# Class Imbalance and Time-To-Detection in the Performance Analysis of Machine Learning-Based Intrusion Detection Systems

**Mohammad Pasha Shabanfar**

**A Thesis**

**in**

**The Department**

**of**

**Concordia Institute for Information Systems Engineering (CIISE)**

**Presented in Partial Fulfillment of the Requirements**

**for the Degree of**

**Master of Applied Science (Information Systems Security) at**

**Concordia University**

**Montréal, Québec, Canada**

**August 2025**

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohammad Pasha Shabanfar**

Entitled: **Class Imbalance and Time-To-Detection in the Performance Analysis of Machine Learning-Based Intrusion Detection Systems**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science (Information Systems Security)**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
*Dr. Arash Mohammadi*

_____Examiner
*Dr. Arash Mohammadi*

_____ Examiner
*Dr. Honghao Fu*

_____ Supervisor
*Dr. Jun Yan*

_____ Co-supervisor
*Dr. Mohsen Ghafouri*

Approved by _____
Chun Wang, Chair
Concordia Institute for Information Systems Engineering (CIISE)

_____ 2025

_____
Mourad Debbabi, Dean
Gina Cody School of Engineering & Computer Science

# Abstract

Class Imbalance and Time-To-Detection in the Performance Analysis of Machine Learning-Based Intrusion Detection Systems

Mohammad Pasha Shabanfar

The increasing reliance on Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) systems has raised critical concerns regarding their vulnerability to cyberattacks. While machine learning (ML) methods have emerged as effective tools for detecting such intrusions, their real-world applicability is challenged by two major issues: the imbalance in cybersecurity datasets and the limited focus on the time required to detect attacks—referred to as Time-To-Detection (TTD). To address these two issues and suggest better practices for ML-based IDS researchers, this thesis examines the gaps in the literature and, through two respective case studies, aims to suggest practices toward a more precise and practical performance assessment of ML-based intrusion detection systems (IDS).

First, the thesis examines the class imbalance problem prevalent in popular Information Technology (IT) and Operational Technology (OT) cybersecurity datasets, where normal traffic often vastly outnumbers attack instances. This imbalance leads to biased model performance and inflated accuracy scores, which can over- or under-assess a model's ability to identify the minority classes correctly. Through a case study with several ML models on a realistic dataset, we demonstrate how imbalanced classes should be considered in the performance evaluation, and how imbalance learning techniques like resampling should be properly utilized for robust performance of ML-based IDS.

Second, this thesis examines TTD, a crucial but often understressed performance indicator that measures how promptly an ML-based detection system identifies the onset of an attack. In addition to traditional metrics that focus solely on classification performance in ML communities, this

thesis proposed a TTD model based on real-world responsiveness in OT systems by defining various stages of the detection process for ML researchers to quantify and measure temporal overheads accordingly. We also demonstrate how the proposed TTD model can be applied to OT datasets through a case study, thereby suggesting it as a best practice for a more comprehensive evaluation of ML-based intrusion detectors in practical OT use cases.

Through the two studies above, the thesis offers a more comprehensive and practical approach to evaluating ML-based IDS. It demonstrates how thoughtful consideration and integration of class imbalance and detection timeliness in the development and assessment of ML-based IDS representation is essential to deploy trustworthy and efficient cybersecurity solutions in critical infrastructure systems.

# Acknowledgments

I sincerely thank everyone who has supported and encouraged me throughout my academic journey. First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Jun Yan, for his continuous guidance, support, and insightful feedback throughout the course of my research. His expertise, patience, and encouragement have been instrumental in shaping the direction and success of this work.

I am also truly grateful to my co-supervisor, Dr. Mohsen Ghafouri, for his valuable advice, technical input, and continuous support. His suggestions and guidance helped me refine my ideas and improve the quality of this thesis.

Most importantly, I would like to thank my parents, whose unconditional love, encouragement, and unwavering belief in me have been the foundation of all my achievements. Their sacrifices, support, and constant motivation have been a source of strength throughout my life. This accomplishment would not have been possible without them.

I would also like to thank my close friends, including Soroush Shahsafi (my cousin, but like a brother to me), Kasra Farrokhi, Pouriya Alikhanifard, Amin Bayatpour, and Pedram Nouri, for always being by my side. Their encouragement, friendship, and support during both challenging and joyful times have been invaluable. A special thanks goes to my dear friend Mohammad Javad Rabiei, whose constant support and kindness throughout this journey have meant a great deal to me.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**API** Application Programming Interface

**CNN** Convolutional Neural Network

**DDoS** Distributed Denial of Service

**DL** Deep Learning

**DNP3** Distributed Network Protocol 3

**DoS** Denial of Service

**EWS** Engineering Workstation

**GPS** Global Positioning System

**GUI** Graphical User Interface

**HMI** Human Machine Interface

**ICS** Industrial Control System

**ICT** Information and Communication Technology

**IDS** Intrusion Detection System

**IED** Intelligent Electronic Device

**IIoT** Industrial Internet of Things

**IoT** Internet of Things

**IT** Information Technology

**KNN** K-Nearest Neighbors

**LSTM** Long Short-Term Memory

**ML** Machine Learning

**MSE** Mean Squared Error

**OMNeT++** Objective Modular Network Testbed in C++

**OT** Operational Technology

**PLC** Programmable Logic Controller

**RAM** Random Access Memory

**ReLU** Rectified Linear Unit

**RNN** Recurrent Neural Network

**ROC** Receiver Operating Characteristic

**RTU** Remote Terminal Unit

**RUS** Random Undersampling

**SCADA** Supervisory Control and Data Acquisition

**SMOTE** Synthetic Minority Over-sampling Technique

**SWaT** Secure Water Treatment

**TCP** Transmission Control Protocol

**TTD** Time-To-Detection

**UDP** User Datagram Protocol

# Chapter 1

# Introduction

## 1.1 Problem Definition

According to the statistics reported for cybersecurity, the damages caused by cyberattacks are expected to reach up to three trillion by 2021, with the probability of executing zero-day exploits one per day [1]. Moreover, the amount of information stored in private and public clouds operated by data-driven companies, such as Amazon Web Services, Facebook, and Twitter, has increased a hundred times by 2022 [1]. The risks of attacks are increasing every day, and critical industrial activities are now directly targeted by cyber attacks (energy and heavy production) [2]. Their identified vulnerabilities, and generally not protected, expose them to significant physical, environmental, and/or financial consequences. According to the "Cisco 2018 Annual Cybersecurity Reports", almost 31% of organizations have been victims of cyberattacks related to the operational technology (OT), while more than 38% predict the occurring of such cyberattacks [2]. 75% of experts give top priority to cyberattack protection, only 15% believe their companies can deal with different cyberattacks [2]. As cyber threats become more frequent and sophisticated, especially in both enterprise and industrial environments, the need for robust protection mechanisms is critical. In particular, network infrastructures and industrial control systems (ICS) have become prime targets due to their increasing connectivity and reliance on newly technologies. This underscores the importance of Intrusion Detection Systems (IDS), which play a vital role in identifying and responding to malicious activities before they can cause significant operational or financial damage.

Machine learning (ML) based IDS provides a learning based system to discover classes of attacks based on learned normal and attack behavior [3]. ML methods are a common solution that is increasingly popular and effective in detecting malware and cyber attacks. The goal of ML-based IDS (based on supervised learning algorithms) is to generate a general representation of known attacks [3]; however, despite their potential, current assessments of ML-based IDS often overlook critical factors such as class imbalance and detection time, leading to evaluations that may not fully reflect real-world performance.

Despite the increasing use of machine learning for intrusion detection, one persistent challenge remains: the class imbalance problem in cybersecurity datasets. In many real-world scenarios, attack instances are significantly fewer than normal traffic, which can lead models to be biased toward majority classes and perform poorly in detecting rare but critical attacks. To mitigate this issue, researchers often apply resampling techniques such as undersampling, oversampling, or hybrid methods to adjust class distributions during training [4]. However, the effectiveness of these techniques can vary depending on how they are applied—particularly whether resampling is performed only on the training data or on both training and testing data. This thesis aims to examine the impact of different resampling strategies on intrusion detection performance and provide insights into best practices for handling imbalanced data in both large-scale and small-scale cybersecurity datasets.

In addition, a critical yet often overlooked challenge lies in accurately evaluating how quickly these systems can detect cyberattacks. The time-to-detection (TTD) metric plays a crucial role in assessing the responsiveness of an IDS; however, existing studies rarely define or measure TTD comprehensively and consistently. Evaluations usually focus on performance-based metrics (e.g., accuracy, precision, recall, F1-Score), without considering the timing of detection in relation to when an attack actually begins. This gap creates uncertainty about the real-world applicability of IDS performance, especially in time-sensitive environments such as ICS, where even milliseconds of delay can lead to serious consequences. As a result, there is a growing need to establish a more accurate and practical approach to defining and measuring true TTD.

## 1.2 Objectives

The main objectives of this thesis are as follows:

- To investigate the impact of various resampling techniques (undersampling, oversampling, and hybrid sampling) on the performance of ML-based intrusion detection systems, with a specific focus on how balancing training data, testing data, or both affects detection accuracy across different dataset sizes.
- Establish a clear and comprehensive definition of the TTD that encompasses the entire duration from the actual start time of a cyberattack to the moment an alert is raised, including all relevant system delays and processing times.

## 1.3 Methodology

This thesis adopts a data-driven experimental methodology to examine the impact of class imbalance and various resampling techniques on the performance of IDSs, specifically using the Convolutional Neural Networks and Long Short-Term Memory (CNN-LSTM) model. The approach focuses on analyzing how balancing strategies applied to training and testing datasets affect detection accuracy, precision, recall, and F1-score, especially in both large-scale and small-scale cybersecurity datasets.

It also defines and measures the TTD of cyberattacks in ICS, with a focus on the Secure Water Treatment (SWaT) dataset. The thesis simulated the relevant delays that happen in the real-world systems for transmitting the messages among tools and devices or time for raising an alert, and implemented the CNN-LSTM model over the SWaT dataset to measure the attack detection by the ML model.

The methodology is structured into the following key steps:

- Conducted a series of experiments using the CNN-LSTM model on CICIDS2017 and SWaT datasets to evaluate the impact of different data balancing methods (undersampling, oversampling, hybrid) under various scenarios (balancing training data, testing data, both, or none),

and analyzed their effects on performance metrics such as accuracy, precision, recall, and F1-score.

- To address the challenge of evaluating detection responsiveness in industrial systems, this thesis introduces a novel model for TTD. The proposed model captures the full timeline, starting from the actual attack initiation to the point at which an alert is raised. This includes key components such as log collection delay, data transmission latency, processing time, and alert generation time.

## 1.4 Contributions

The main contributions of this thesis are as follows:

- Conducting a comprehensive literature review on machine learning-based intrusion detection systems (ML-based IDS), identifying key gaps in current evaluation practices—particularly regarding class imbalance and detection time—as two critical aspects that require more rigorous assessment methodologies.

- Performing a comparative case study on the impact of class imbalance in both training and testing datasets, using IT and OT benchmark datasets, to empirically demonstrate its effect on model performance and provide practical recommendations for more reliable evaluation of ML-based IDS.

- Proposing a new TTD model tailored for ICS environments and applying it in a case study using the SWaT dataset and a CNN-LSTM-based IDS, thereby illustrating overlooked delays in detection pipelines and enabling more realistic benchmarking of intrusion detection responsiveness.

## 1.5 Thesis Structure

The thesis is structured into six chapters, each addressing a specific aspect of the research. The second chapter reviews the related works in the field of IDS for ICSs. It begins by explaining the

role of IDSs in enhancing cybersecurity, then categorizes public cybersecurity datasets based on their testbed environments into Information Technology (IT) and Operational Technology (OT) domains. This classification is followed by an introduction to several widely used IT and OT datasets, highlighting their characteristics and suitability for different detection use cases.

Chapter three presents an in-depth case study on the impact of class imbalance in cybersecurity datasets and evaluates various data resampling techniques. This chapter investigates how undersampling, oversampling, and hybrid sampling influence the performance of deep learning models under different conditions where either the training data, testing data, or both are balanced or imbalanced. The experiments are conducted on two benchmark datasets—CICIDS2017 and SWaT—to provide a comprehensive analysis across large- and small-scale scenarios. The results are analyzed to offer insights into which resampling strategies are most effective in practical intrusion detection settings and to address the implications of balancing choices on model generalization and detection performance.

The fourth chapter presents a proposed TTD model, describing the detection timeline from the start of an attack to the final alert signal. The model is developed to address limitations in current detection time measurements and to define each component of the detection process with clarity, making it suitable for empirical evaluation. It details the implementation and evaluation of the proposed TTD model using the SWaT dataset. A hybrid CNN-LSTM model is implemented to detect attacks, and the timestamps of attack detection are recorded.

The final chapter concludes the thesis by summarizing the findings, highlighting the contributions made to the field, and discussing the implications and potential directions for future work in measuring and optimizing detection time in cybersecurity systems for ICSs.

# Chapter 2

# Literature Review

## 2.1 IT And OT

### 2.1.1 IT Security in ICS

IT refers to systems that manage data, communication, and computing resources in business and enterprise environments. It includes technologies like computers, servers, databases, web applications, cloud platforms, and email systems.

Security in IT systems is understood conceptually in the academic literature and to a degree in practice in the enterprise environment. IT security measures have evolved over the past two decades from a binary "secure or not secure" measurement to one based on risk. Since risk management is already a functioning business requirement, the risk management concept has made it easier to integrate security into business decisions. The highest priority in IT systems is confidentiality of data [5].

IT security is widely used on the public Internet as well as ICS for many applications, e.g., email, voice-over-IP, transportation, healthcare, and many other sectors [5]. These networks facilitate internal and external communication for employees, suppliers, and customers. Backend offices of energy companies are another example of maintaining corporate IT networks that handle administrative tasks, finance, human resources, and other business operations.

### 2.1.2 OT Security in ICS

OT involves hardware and software used to monitor and control physical processes in industries like energy, water, manufacturing, and transportation. It includes systems such as Supervisory control and data acquisition (SCADA), Programmable Logic Controller (PLC), sensors, actuators, and control networks.

These systems enable computers to manage operational procedures and make data accessible to the business [5]. OT communications are widely used in the ICS at present. SCADA systems are used to monitor and control industrial processes and infrastructure. They use various OT communication protocols to gather data from sensors and control equipment. Distributed Control Systems (DCSs) are used in manufacturing and process industries to control and automate production processes. They rely on dedicated communication networks for real-time control [6].

OT systems follow two main categories of security specifications: one derived from general-purpose IT security practices, and another shaped by the unique requirements of infrastructure segmentation and industrial operations [5]. Security standards and recommendations in OT environments often vary depending on the sector, business context, and criticality of the systems involved, making some frameworks more applicable than others [5]. In OT systems, the availability of data has the highest priority. At the core of many industrial OT environments are PLCs, which regulate machinery by receiving sensor inputs and sending control commands. These devices rely on specialized OT communication protocols tailored for real-time control and safety. One such protocol is Generic Object-Oriented Substation Events (GOOSE) [7], which is part of the IEC 61850 standard suite. GOOSE facilitates high-speed communication of status events in electrical substations and is commonly used in substation automation and protection systems [8].

### 2.1.3 Comparison of IT Security with OT Security in ICS

The importance of OT security is as well as IT security. While the systems may not be fully developed in terms of technological security capabilities, they are well-regulated from a regulatory standpoint. The focus of IT security is on protecting information, networks, and computer systems. In contrast, OT systems are related to the control and automation of physical processes, such as

manufacturing, industrial machinery, and critical infrastructure. Therefore, even though OT systems (i) may not have many technical level controls, such as access control systems and cryptography, and (ii) though they may not have much to no forensic and logging capability, these features are all governed by regulatory decree. OT personnel are placed in a cognitive cage where regulatory compliance takes precedence over security considerations when regulatory edicts are used instead of robust security functionality. Cognitively, OT employees may confuse security with regulatory compliance, which is difficult to spot before a significant preventable incident. Figure 2 illustrates some differences between IT and OT in terms of priority, risks, networks, and protocols [9].

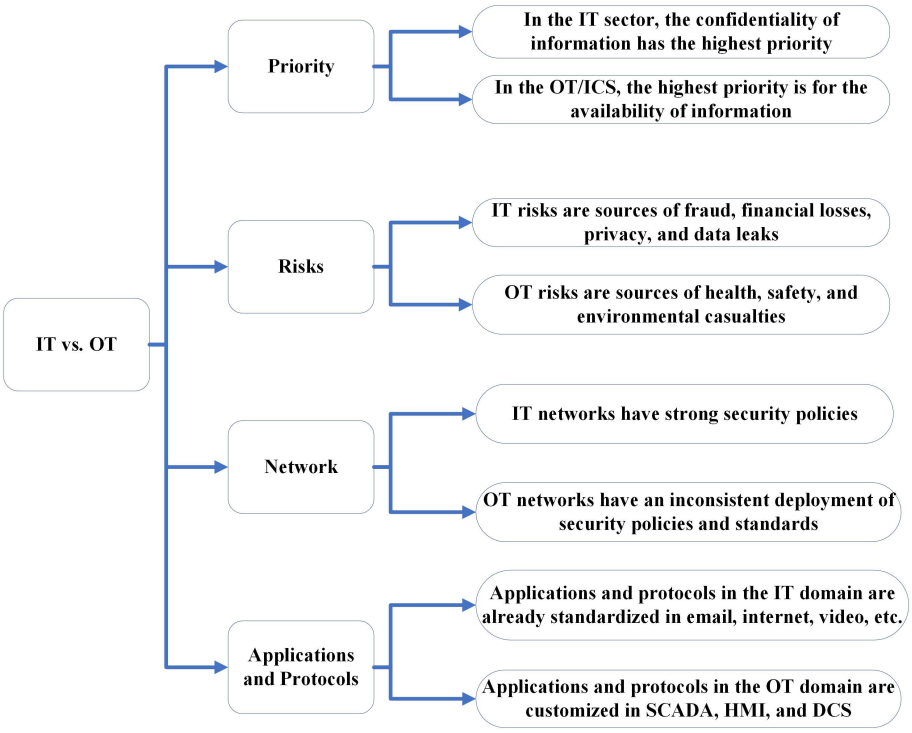


Figure 2.1: Differences between IT and OT in ICS security.

## 2.2 IDS For ICS Security

ICS are essential components of critical infrastructure, including power grids, water treatment facilities, and manufacturing plants. These systems are increasingly becoming targets for cyber-attacks due to their connectivity with corporate networks and the internet [10]. Unlike traditional

Information Technology (IT) systems, ICS environments have unique security requirements due to their real-time operational constraints, legacy devices, and deterministic communication protocols [11].

The ICS experimental environment generally consists of three levels, which have been shown in Figure 2.2 [12]. Devices should be located and set up during the environment's construction. Additionally, a system for collecting various data is implemented during ICS operation. Level 0 represents the physical process layer, comprising sensors, pumps, actuators, and breakers that directly interact with industrial operations. Level 1 serves as the control layer, hosting devices such as Programmable Logic Controllers (PLCs), Distributed Control Systems (DCS), Remote Terminal Units (RTUs), and Intelligent Electronic Devices (IEDs), which process field data and issue control commands. Level 2 functions as the supervisory layer, including components like Engineering Workstations (EWS), Human-Machine Interfaces (HMI), data Historians, and GPS Clocks. These systems facilitate operator interaction, data visualization, and time synchronization. Communication across and within layers is enabled via wired and wireless connections through dedicated network equipment. This layered design is essential for structuring control logic, ensuring reliability, and providing a foundation for security monitoring and response.

Security issues related to computers have been increasingly critical and must be tackled consistently. In the computer security community, an important component for defending a computer system is the IDS. IDS are centered on the assumption that intruders exhibit behavior that is dissimilar from the user's normal behavior [13].

Based on deployment type, IDSs can generally be divided into three groups: host-based, network-based, and hybrid [14]. An application program placed on a host computer that may examine and track system behavior is known as a host-based IDS (HIDS). HIDS is particularly useful for identifying insider threats, privilege escalations, or malware operating at the host level [15]. A Network-Based Intrusion Detection System (NIDS) observes and analyzes traffic across an entire network to detect potential intrusions. It is typically deployed at key points within the network, such as gateways, routers, or between subnets, where it inspects packets in real time to identify known attack patterns or abnormal behaviors [16]. Hybrid Intrusion Detection System combines the capabilities

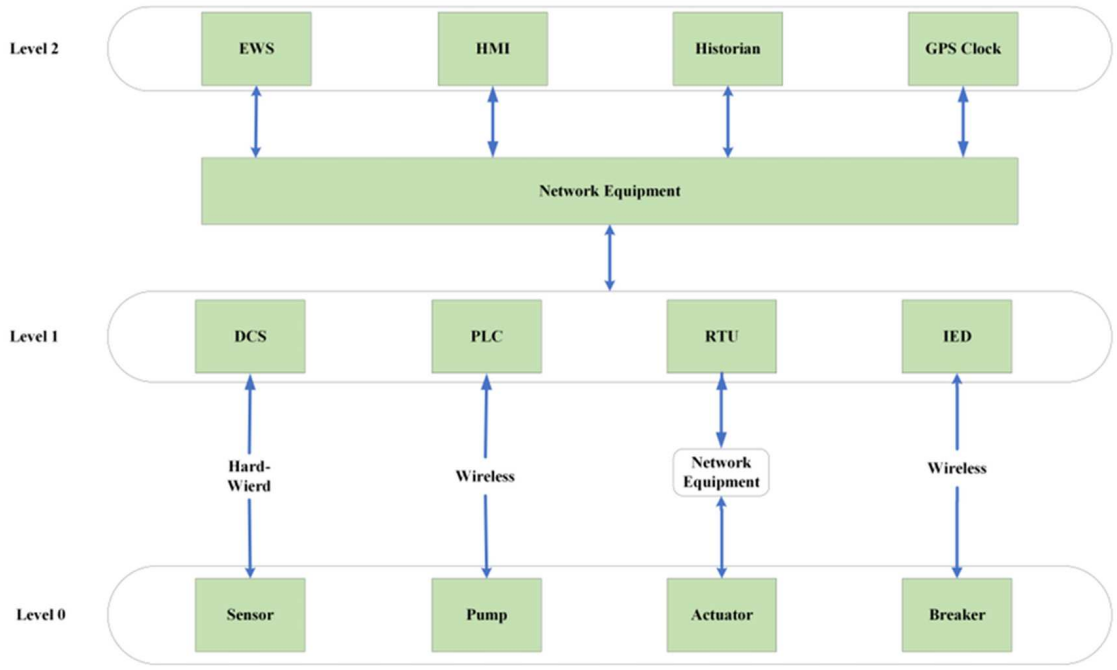


Figure 2.2: Example of an ICS environment.

of both HIDS and NIDS to provide a more comprehensive defense mechanism. By correlating host-level insights with network-wide observations, hybrid IDS solutions can detect complex, multi-stage attacks that might go unnoticed when using either method alone [17].

Based on the detection approach, IDSs can be divided into anomaly-based and signature-based. Anomaly-based intrusion detection distinguishes between malicious and typical patterns by correlating with typical behavior models. Signature-based IDS operates by comparing network traffic against a database of known attack patterns. This approach is highly effective in detecting well-documented threats, such as malware and exploits with known signatures [18]. Tools such as Snort and Suricata are widely used in ICS environments for their ability to match traffic against predefined rulesets [19].

The signature of the attack is kept in the database in a signature-based IDS. Signature-based IDS compares the signature contained in the database to the attack signature and triggers an alarm if any mismatch is detected when breaking the invading endeavor. When comparing anomaly and signature detection, anomaly detection outperforms signature detection because signature detection identifies only known assaults, but anomaly detection detects both known and undiscovered threats.

Stateful protocol recognition methods match the identified activities and detect the deviation from the state of the protocol [14]. Currently, anomaly-based IDS and signature-based detection methods are commonly used for IDS, but these two detection methods have a disadvantage, such as a high false positive rate and a lower detection rate [20]. Artificial intelligence (AI)-based detection techniques have gained attention in IDS research due to their effectiveness in IDS [21].

## 2.3 ML-Based IDS for ICS Security

The adoption of ML-Based Intrusion Detection Systems (ML-IDS) in ICS has gained significant attention due to their ability to detect unknown threats and adapt to evolving attack patterns. Unlike traditional IDS approaches, ML-IDS leverages data-driven techniques to identify malicious behavior by analyzing patterns and relationships in network traffic and system logs[22]. ML-based IDS can be categorized into supervised, unsupervised, and semi-supervised learning models, each offering unique advantages and challenges in ICS environments.

Supervised learning-based IDS relies on labeled datasets to train models that can distinguish between normal and malicious activities. Algorithms, such as Support Vector Machines (SVM), Decision Trees, Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM), and Neural Networks (NN) are widely used for classification tasks. However, the effectiveness of supervised models is heavily dependent on the quality and diversity of labeled datasets, which are often scarce in ICS environments [23].

Unsupervised learning requires no human intervention because it learns by grouping similar data together to form clusters or associations. This type of learning is desirable when labels are absent or insufficient from the training data. Common unsupervised learning algorithms found in the literature for ICS attack detection are Isolation Forest, OneClass Support Vector Machine (OCSVM), and Autoencoders such as Sparse Autoencoders (SpAE), Undercomplete Autoencoders (UAE), Variational Autoencoders (VAE), and Fair Clustering (FD)[24].

Semi-supervised learning combines a small set of labeled data with a large amount of unlabeled data to improve model performance, especially when labeling is expensive or time-consuming. It is useful in intrusion detection for ICS, where annotated data is often limited. Semi-supervised

generative models like VAEs enable the model to learn from both types of data, improving accuracy and adaptability in detecting novel attacks [25].

## 2.4 Datasets for ML-based IDS in ICS Security

### 2.4.1 IT Datasets for ML-based IDS in ICS

An IT dataset can be developed by collecting information from varied sources, such as network traffic flows that contain information about the host, user behaviour, and system configurations [26]. This information is required to study various network attack patterns and abnormal activity. The network activity is collected through a router or network switch. After collecting the incoming and outgoing network traffic, network flow analysis is performed to study the traffic. Flow analysis involves analyzing network packet information, including source IP address, destination IP address, source port number, destination port number, and type of network services, among others. The network host delivers the system configurations and user information that cannot be extracted from the network flow analysis [27].

According to the categorization, some public IT datasets frequently used for intrusion detection in ICS are:

**CICIDS 2017**. CICIDS 2017 [28], generated over five days in an emulated environment, encompasses network traffic presented in both packet-based and bidirectional flow-based formats. The dataset comprises extensive attributes, exceeding 80 for each flow, accompanied by supplementary metadata concerning IP addresses and attack details. It encompasses a wide range of attack types, including but not limited to SSH brute force, Heartbleed, botnet, Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), web, and infiltration attacks; however, the dataset is imbalanced over all kinds of attacks.

**CIC DOS**. The CIC DoS dataset [29] from the Canadian Institute for Cybersecurity was developed to construct an intrusion detection dataset featuring application-layer DoS attacks. To achieve this, the researchers conducted eight distinct application-layer DoS attacks. To generate normal user behavior data, they merged the obtained traces with attack-free traffic extracted from the ISCX 2012 dataset.

**DDOS 2016** . The DDOS 2016 dataset [30], constructed in 2016 using the NS2 network simulator, adopts a packet-based format. Unfortunately, specific details regarding the simulated network environment remain undisclosed. Within the DDoS 2016 dataset, attention is primarily directed toward various categories of DDoS attacks. In addition to normal network traffic, this dataset encompasses four distinct types of DDoS attacks: UDP flood, Smurf, HTTP flood, and SIDDOS. Although it contains both normal and attack traffic, its primary focus is on simulating various types of DDoS attacks, which often results in a disproportionate number of attack packets relative to normal packets.

**UNSW-NB 15** . The UNSW-NB15 dataset, as outlined in [30], comprises both regular and malicious network traffic, presented in a packet-based format. This dataset was generated within a confined emulated environment over a period of 31 hours, utilizing the IXIA Perfect Storm tool. It encompasses a diverse array of attack categories, including but not limited to backdoors, DoS, exploits, fuzzers, and worms, forming nine distinct attack families. UNSW-NB15 is equipped with predefined partitions for training and testing purposes, featuring a total of 45 unique IP addresses in the dataset.

**DARPA, KDD CUP, NSL-KDD**. The DARPA 1998/99 datasets [31], widely recognized as the primary datasets for intrusion detection, were crafted at the MIT Lincoln Lab in an emulated network environment. Comprising packet-based network traffic data, the DARPA 1998 dataset spans seven weeks, while the DARPA 1999 dataset covers five weeks.

KDD CUP 99 [32], derived from the DARPA 98 dataset, ranks among the most extensively employed datasets for intrusion detection purposes. This dataset includes fundamental attributes related to TCP connections and higher-level features, such as the count of unsuccessful login attempts, although it omits IP addresses.

NSL-KDD [33], an evolved dataset, was created as a response to duplicate data concerns within KDD CUP. This dataset, derived from the original KDD Cup99 dataset, was created based on Tavallaee et al.'s analysis of the KDD training and test sets, which revealed that duplicate network packets accounted for approximately 78% and 75% of both the training and testing subsets.

A detailed summary of IT data sets is shown in Table 2.1. According to Table 2.1, general information on IT datasets, such as normal and attack data, the amount of the datasets, and their

Table 2.1: General Information of IT Datasets

| Dataset | Normal Traffic | Attack Traffic | Anonymity | Count | Duration | Kind of Traffic |
|---|---|---|---|---|---|---|
| CIC DoS | yes | yes | none | 4.6GB packets | 1 day | emulated |
| CICIDS 2017 | yes | yes | none | 3.1M flows | 5 days | emulated |
| DARPA | yes | yes | none | not specified | 7.5 weeks | emulated |
| DDoS 2016 | yes | yes | yes | 2.1M packets | none specified | synthetic |
| KDD Cup 99 | yes | yes | none | 5M points | none specified | emulated |
| NSL-KDD | yes | yes | none | 150K points | none specified | emulated |
| UNSW-NB15 | yes | yes | none | 2M points | 31 hours | emulated |

kind of traffic, has been shown. Moreover, most datasets have been generated in an emulated environment, and there are fewer datasets with real network environments. The presence of specific attack scenarios is an important aspect when searching for a network-based data set. According to [33], which describes specific attacks within IT datasets, DoS, DDoS, port scans, and botnets are the most prevalent attacks used in the datasets to simulate abnormal network situations.

### 2.4.2 OT Datasets for ML-based IDS in ICS

This section discusses public OT datasets used in several surveys to detect attacks by implementing different algorithms and methods. ICSs are one of the most effective tools to prevent cyberattacks. The key components of the ICS include SCADA, HMI, PLC, RTU, and DCS. A SCADA system helps collect data from field sensors, enabling us to control the system through HMI software [9]. OT monitors all industrial systems, and ICS relates to the security of industrial systems. Thus, ICS datasets are a type of subset of OT datasets.

Here are some publicly available OT datasets frequently used for detecting algorithms, which we will compare based on four categories discussed in the next subsection.

**Morris et al. Datasets [12].** For their research on intrusion detection, Morris et al. have made five separate datasets about the production of electricity, gas, and water available. The Morris datasets can be used for ML in creating intrusion detection systems because they all provide labels in common. The Morris-1 dataset includes 37 scenarios for power system events that consider the number of IED operations and typical and unusual occurrences in the testbed for power systems comprising generators, IEDs, breakers, switches, and routers. The RS-232, or Ethernet interface

in the gas pipeline testbed, is connected in the Morris-2, Morris-3, and Morris-4 datasets to enable Modbus protocol connection between the control device and the HMI. Every dataset has network data information that has some header information removed.

**Lemay [12]**. Lemay et al. have contributed a network traffic dataset focused on covert channel command and control within the SCADA domain. To create the testing environment, a SCADA network was established using the publicly available SCADA Sandbox tool. Additionally, two master terminal units were implemented through SCADA. The dataset encompasses Modbus/TCP communication, involving the connection of three controllers and four field devices per controller.

**Rodofile et a. Dataset [12]**. It comprises two elevated reservoir tanks, six consumer tanks, two raw water tanks, and a return tank. It contains chemical dosing systems, booster pumps, valves, instrumentation, and analyzers. WADI is controlled by 3 PLCs that operate over 100 network sensors. Moreover, the testbed is equipped with a SCADA system. WADI consists of three main processes: (i) P1 (Primary supply and analysis), (ii) P2 (Elevated reservoir with Domestic grid and leak detection), and (iii) P3 (Return process). Its use cases are to demonstrate that the detection mechanism applies to real-world ICS data and to determine whether any attack methodology is transferable from a scenario using simulated data to another scenario using real data.

**SWaT [12]**. The SWaT dataset encompasses sensor data, actuators, PLC input/output (I/O) signals, and network traffic, which were recorded over a duration of four days during an assault scenario and seven days under regular operational conditions. The dataset was generated from a real-world water treatment testbed developed at the Singapore University of Technology and Design (SUTD). This testbed simulates a six-stage water purification process, which includes chemical dosing, filtration, and storage. It is worth noting that the SWaT datasets represent one of the most extensive data collections within a substantial testbed. SWaT has meticulously crafted a total of 36 attack scenarios, encompassing both field signals and network traffic; however, it does not provide precise timestamps for attack initiation and alert generation.

**WADI [34]**. The WADI testbed comprises a comprehensive facility encompassing two elevated reservoir tanks, six consumer tanks, two raw water tanks, and a return tank. Within this setup, you'll find an array of essential components, including chemical dosing systems, booster pumps, valves, instrumentation, and analyzers. The control of WADI is managed by three PLCs, each

communicating with over 100 network sensors. Additionally, the testbed is equipped with a SCADA system to facilitate monitoring and control.

**EPIC [35]**. Data from the EPIC testbed encompasses eight distinct scenarios during normal operation, each scenario spanning approximately 30 minutes. The data collected includes sensor and actuator information, meticulously recorded in an Excel spreadsheet, and network traffic data, which has been archived in *.pcap* files. EPIC represents a power testbed that faithfully replicates a compact real-world smart grid system, encompassing four essential stages: generation, transmission, microgrid, and smart home. Each stage is under the control of its dedicated PLC/controller. Additionally, communication channels are established between the SCADA system, the DCS, the energy management system (EMS), and each PLC/controller.

**WUSTL [36]**. This dataset encompasses network data derived from the Industrial Internet of Things (IIoT) for the purpose of cybersecurity research. The primary objective of this testbed is to replicate real-world industrial systems with maximum fidelity, enabling the execution of genuine cyber-attacks for research purposes. A substantial volume of data, totaling 2.7 GB, was accumulated over a period of approximately 53 hours, which was preprocessed and cleaned for research purposes

Here, we briefly compare the OT datasets. We compare datasets based on their public information, data domain, and size of normal and attack data. Each dataset is collected from its own experimental environment in a specific or complex domain. To specify our analysis target, we limited our study to the ICS-related datasets that can be accessed publicly. Table 2.2 describes the data domain and size of the normal and attack of some of the datasets as an example.

Table 2.2: Data Domain and Dimensions of Normal and Attack of OT Datasets

| **Dataset** | **Data Domain** | **Num. of Normal Traffic (%)** | **Num. of Attack Traffic (%)** |
|---|---|---|---|
| Morris5 | EMS | 16,362(92.09) | 1,405(7.91) |
| Lemay | SCADA | 395,298(87.86) | 54,321(12.14) |
| Rodofile | Mining Refinery | 1,137,294(63.09) | 665,463(36.91) |
| SWaT | Water Treatment Plant | 395,298(87.85) | 54,621(12.5) |

## 2.5 Assessments of ML-based IDS for ICS Security

While various public datasets such as CICIDS2017, SWaT, and NSL-KDD have been widely used for benchmarking IDSs, they exhibit notable limitations that hinder their effectiveness in industrial cybersecurity research.

Many datasets, especially IT datasets, lack essential content that is critical for evaluating time-based detection metrics. In particular, they often do not provide precise timestamps for attack initiation and alert generation. This absence makes it difficult or even impossible to assess TTD, a crucial performance indicator for real-time intrusion detection in ICS environments. The inability to align attack events with detection outcomes limits the value of such datasets for evaluating the temporal responsiveness of IDS solutions.

Also, class imbalance remains a persistent issue. Several widely used datasets contain a disproportionately small number of attack samples compared to normal traffic, leading to biased model performance and unreliable evaluation results. Additionally, few datasets offer configurable versions that allow researchers to explore different sampling ratios or imbalance scenarios systematically.

# Chapter 3

# Consideration Class Imbalance in ML-based IDS

## 3.1 Intoroduction

The class imbalance problem occurs when one or more classes in a dataset contain significantly more instances than other classes. This imbalance causes ML models to focus disproportionately on the majority class, often resulting in the misclassification of the minority class [37]. This issue is particularly problematic when the minority class holds critical importance, such as in the detection of malignant cancer samples, fraud events, or network intrusions.

Imbalanced datasets are an important issue in both binary and multi-class classification problems. A binary class is a problem in which one of two classes has more instances than the other. In a binary classification, the goal is to distinguish between two distinct classes. In the context of cybersecurity, binary classification refers to the distinction between normal and attack classes. When one of these two classes contains substantially more samples than the other, the class with more instances is referred to as the majority class, while the one with fewer instances is called the minority class.

In contrast, a multi-class classification problem involves three or more classes. It is not necessarily that class imbalance in this context points to a single class being under-represented; multiple

classes may have significantly fewer samples compared to others, or there can be an uneven distribution across all the classes [38]. This enhances the complexity in dealing with multi-class imbalance, as the model must distinguish between multiple classes that are not equally represented, which can bias predictions in favor of the majority classes.

The deterioration of a given model may go unnoticed if the method is only evaluated by metrics such as accuracy [4]. This highlights the importance of selecting evaluation metrics beyond accuracy to address the imbalance challenge effectively, ensuring that models can reliably identify instances of the minority class..

In cybersecurity, the minority classes often correspond to rare but critical attack types. For example, ICS datasets may have a higher volume of benign traffic compared to rare but significant attacks, such as SQL injection or advanced persistent threats. Addressing this imbalance is crucial to improving detection rates for these critical events.

Several approaches have been developed to mitigate the class imbalance problem in terms of cybersecurity. Various methods to counter that issue have been proposed. These fall roughly into three categories: undersampling, oversampling, and cost-sensitive methods [4]. Sampling techniques, such as duplicating samples from the minority class (oversampling) or reducing samples from the majority class (undersampling), are commonly employed to rebalance datasets. However, these methods have limitations, such as the potential for overfitting in oversampling or the loss of important information in undersampling [38]. Cost-sensitive approaches, on the other hand, assign higher penalties for misclassifying instances of the minority class, making them an effective algorithm-level solution. Despite advancements, the challenge of imbalanced datasets persists, particularly in multi-class problems, where ensuring an equitable focus across all classes is more complex.

Real-world datasets come from actual industrial operations and often exhibit significant imbalances due to the rarity of cyberattacks. These datasets are captured directly from ICS/OT environments during their regular operation. The inherent imbalance arises from the predominance of normal traffic (benign data) compared to the infrequent occurrence of malicious traffic (e.g., anomalies or cyberattacks). This severe class imbalance makes it challenging for ML models to effectively identify and classify rare attack instances among the overwhelming number of normal operations.

Within ICS, this difference between normal and malicious traffic is usually more stark, due to the highly deterministic nature found within OT networks. In contrast to IT networks, where varied user behaviour produces disparate traffic flows, ICS traffic is characterized by repetitive and predictable communication between devices, including programmable logic controllers (PLCs), remote terminal units (RTUs), and supervisory control and data acquisition (SCADA) systems. Hence, normal traffic vastly predominates, whereas cyberattacks are extremely infrequent, resulting in an enormous class imbalance. Additionally, OT-specific threats, such as command injection, sensor spoofing, and protocol-based attacks, occur at varying frequencies depending on the system architecture and security posture. This skew not only affects detection models but also biases performance evaluation metrics because traditional accuracy-based metrics can be misleading in such an environment. Thus, an accurate description of imbalance in the ICS dataset must consider the nature of control traffic and the frequency distribution of different attack types, which vary by industry sector, network topology, and the sophistication of attacks.

Attack categories themselves are also imbalanced in real-world datasets. For instance, common attacks like Denial of Service (DoS) may appear more frequently, while sophisticated and highly targeted attacks, such as Stuxnet-like incidents, are exceedingly rare or even absent. Additionally, ICS/OT datasets often involve high-dimensional data, where dominant features related to regular operations may overshadow the subtle signatures of cyberattacks. This high dimensionality and imbalance further complicate the task of accurately detecting and analyzing anomalies in real-world datasets.

On the other hand, open-access synthetic datasets used for ICS cybersecurity often exhibit varying degrees of class imbalance, depending on their design purpose and target use cases. While real-world datasets are inherently imbalanced due to the rarity of cyberattacks, synthetic datasets are generated to address this imbalance, ensuring sufficient representation of malicious traffic for effective ML training and evaluation.

Existing synthetic datasets typically exhibit moderate imbalance ratios, ranging from balanced (50:50) to moderately imbalanced (80:20 or 90:10), depending on the dataset's objective. For example, datasets like NSL-KDD and CICIDS 2017 have an imbalance ratio of 80:20 between benign and attack data, which is appropriate for implementing ML methods [39]. This balanced design

enhances model generalization across various attack types, although it may not fully capture the rarity of specific attacks in operational environments.

### 3.1.1 Imbalance Ratio of Known Datasets

The imbalance ratio, calculated using Equation (1), is a widely used metric to quantify class imbalance in datasets [39]. The equation is expressed as:

$$\text{Imbalance Ratio} = \frac{\max_i\{C_i\}}{\min_i\{C_i\}} \tag{1}$$

In this equation, $C_i$ represents the data size of class $i$, with $\max_i\{C_i\}$ indicating the size of the largest class and $\min_i\{C_i\}$ representing the size of the smallest class. A higher imbalance ratio signifies a greater disparity between class distributions, which can pose significant challenges for ML models. For instance, models trained on datasets with high imbalance ratios often struggle to identify minority class samples, as they are overshadowed by the majority class [39].

Table 3.1: Dataset Distribution Across Classes.

| Dataset | Benign | Bot | Brute Force | DoS | Infiltration | SQL Injection |
|---|---|---|---|---|---|---|
| KDD Cup 99 | 4,113,223 | 553,301 | 45,268 | 18,599 | 112 | - |
| NSL-KDD | 77,054 | 53,387 | 14,077 | 4,833 | 119 | - |
| CIC-IDS2017 | 2,358,036 | 453,438 | 15,967 | 1,966 | 36 | 21 |
| CSE-CIC-IDS2018 | 2,856,035 | 1,289,544 | 286,191 | 93,063 | 513 | 53 |

Table 3.1 shows the number of records of the most preferred and popular datasets, which are categorized by their classes. As can be seen, these datasets are not balanced. According to Equation (1), the imbalance ratio of datasets is listed in Table 3.2 [39].

The imbalance ratio of datasets significantly influences the performance evaluation of ML models, particularly in cybersecurity for ICS. Imbalanced datasets can distort commonly used evaluation metrics such as accuracy. For example, if 95% of the data is benign and only 5% represents attacks, a model predicting all samples as benign would achieve 95% accuracy while failing to detect any attacks. Thus, metrics like precision, recall, F1 score, and area under the ROC curve (AUC-ROC) are more reliable for assessing performance under imbalance conditions.

Models trained on imbalanced datasets often exhibit poor detection of minority-class instances, which correspond to malicious traffic in ICS datasets. This results in high false-negative rates, where critical attacks might go undetected, compromising system security. Moreover, imbalance affects detection time, a crucial metric for ICS security. Models struggling to identify minority-class instances often exhibit delays in recognizing attacks, reducing the effectiveness of real-time threat mitigation [40].

Table 3.2: Imbalance Ratio For Each Dataset.

| **Dataset** | **Imbalance Ratio (x:1)** |
|---|---|
| KDD Cup99 | 36,725.20 |
| NSL-KDD | 647.51 |
| CIC-IDS2017 | 112,287.42 |
| CSE-CIC-IDS2018 | 53,887.45 |

## 3.2 Generic Resampling Techniques

Resampling methods are important for improving the effectiveness of ML models, especially when dealing with imbalanced datasets. In datasets with high skewness, such as SWaT and CIC-IDS2017, where attack instances are significantly less common than normal instances, models tend to exhibit a bias towards the majority class, resulting in poor detection of attacks from the minority class. Resampling methods offer a solution to this problem by either increasing the proportion of minority samples (oversampling) or reducing the number of majority samples (undersampling).

Under-sampling balances the dataset by decreasing the size of the majority class. This method is adopted when the number of elements belonging to the majority class is rather high. In that way, one can keep all the samples belonging to the minority class and randomly (or not) select the same number of elements representing the majority class [4]. Oversampling, on the other hand, increases the minority class's percentage by randomly reproducing it [41]. Figure 3.1 shows the overview of resampling methods.

Resampling techniques play a pivotal role in addressing the problem of imbalanced datasets by altering the ratios between majority and minority classes. These techniques aim to provide

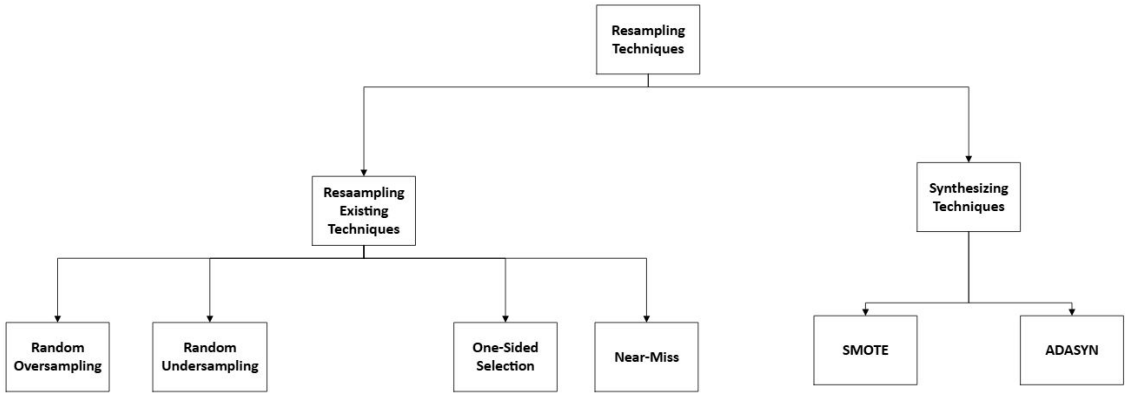


Figure 3.1: Explanation of Resampling Techniques

balanced data for training ML models, thereby improving their performance. Below, various resampling methods, such as Random Undersampling (RUS) [42], Random Oversampling (ROS) [42], Synthetic Minority Oversampling Technique (SMOTE) [43], and Adaptive Synthetic Sampling (ADASYN) [43], are discussed along with their advantages and disadvantages.

### 3.2.1 Existing Resampling Techniques For IDS

#### Random Oversampling (ROS)

Random Oversampling [42] increases the number of minority class samples by duplicating existing samples through random selection with replacement. The mechanics of random oversampling follow naturally from its description by adding a set $E$ sampled from the minority class: for a set of randomly selected minority examples in $S_{\text{min}}$, augment the original set $S$ by replicating the selected examples and adding them to $S$. In this way, the number of total examples in $S_{\text{min}}$ is increased by $|E|$ and the class distribution balance of $S$ is adjusted accordingly [44]. While ROS helps balance the dataset and improves the representation of the minority class, it can lead to longer training times due to the larger dataset. Furthermore, the duplication of data may increase the risk of overfitting, as the model might fail to generalize well to unseen data.

### Random Undersampling (RUS)

Random Undersampling [42] involves reducing the number of samples in the majority class by randomly selecting a subset of data without replacement. While oversampling appends data to the original dataset, random undersampling removes data from the original dataset. In particular, we randomly select a set of majority class examples in $S_{\text{max}}$ and remove these samples from $S$ so that $|S| = |S_{\text{min}}| + |S_{\text{max}}| - |E|$. Consequently, undersampling readily gives us a simple method for adjusting the balance of the original dataset $S$ [44]. This technique significantly decreases the training time due to the smaller dataset size. However, RUS carries the risk of removing important information from the majority class, which may lead to reduced model accuracy and higher misclassification rates. Despite its simplicity and effectiveness in reducing computational costs, the potential information loss limits its broader applicability .

### One-sided Undersampling

One-sided selection [45] proposes a removal of cases belonging to the majority class while leaving untouched all cases from the minority class, because such cases are rare and can be lost. The one-sided selection removes redundant and borderline instances using Tomek links and nearest neighbours. A Tomek link exists between two points $x_i$ and $x_j$ if:

$$d(x_i, x_j) = \min_{x_k \in D} d(x_i, x_k) \quad \text{and} \quad d(x_j, x_i) = \min_{x_k \in D} d(x_j, x_k) \tag{2}$$

where $d(x_i, x_j)$ is a distance metric (e.g., Euclidean distance). It removes instances from the majority class that form Tomek links with the minority class. The one-sided selection technique aims to create a training set consisting of safe cases. One-sided undersampling differs from random undersampling in that it removes representative majority-class instances that are in favor of the decision boundary, rather than any random majority-class sample. This approach generally involves identifying the majority samples that are closest to the minority samples in the feature space, while simultaneously eliminating redundant or less informative examples. To achieve this, noisy, borderline, and redundant majority-class cases should be eliminated.

#### Near-Miss Undersampling

Near Miss [46] is a family of under-sampling techniques that remove the majority of samples based on their average distances from the minority class. NearMiss selects the majority class samples closest to the minority class samples using $k$-nearest neighbors (KNN):

$$D' = \{x_i \mid x_i \in D_m, x_i \in \text{KNN}(D_s)\} \cup D_s \tag{3}$$

where $D'$ is the undersampled dataset, and $\text{KNN}(D_s)$ is the set of majority class points closest to the minority class. Depending on the nearest neighbour algorithms used to measure the distance, three Near-Miss methods are proposed. Near Miss-1 selects the majority samples with the smallest average distance to the three closest samples from the minority class. Near Miss-2 selects the majority samples with the smallest average distance to the three farthest samples from the minority class. This method guarantees that the boundary between classes is clearly defined, thereby ensuring less loss of information compared to random undersampling. Near Miss-3 selects a fixed number of nearest majority class samples for each minority class sample. Nonetheless, the Near-Miss method can result in increased training time and overfitting to a certain part of the majority class.

### 3.2.2 Synthetic Sampling

#### Synthetic Minority Oversampling Technique (SMOTE)

SMOTE [43] is a data augmentation technique used to increase the data from minor classes. It synthesizes new data instances by randomly selecting a new data point near k neighbours that belong to a minor class. SMOTE generates synthetic samples between existing minority class samples. Given a minority class sample $x_i$ and its nearest neighbour $x_j$, a synthetic sample $x_{\text{new}}$ is generated as:

$$x_{\text{new}} = x_i + \lambda \cdot (x_j - x_i) \tag{4}$$

where $\lambda \sim U(0, 1)$ is a random number. SMOTE overcomes the limitations of ROS by avoiding simple duplication and instead generating diverse synthetic data points. This reduces the risk of

overfitting while enhancing model performance, particularly in tasks involving imbalanced datasets, such as network intrusion detection. However, SMOTE requires at least three samples in the minority class to function effectively, limiting its applicability to extremely small datasets. Additionally, SMOTE does not consider class boundaries and may generate synthetic data points that are far from decision boundaries, which may fail to improve classification accuracy in critical regions. Moreover, the increase in training time due to additional synthetic samples remains a disadvantage.

**Adaptive Synthetic Sampling (ADASYN)**

ADASYN [43] is another resampling method that focuses on generating synthetic samples for minority-class instances that are harder to classify. ADASYN assigns different weights to synthetic samples based on the local density of instances from the minority class. The number of synthetic samples for each minority class instance $x_i$ is:

$$G_i = \frac{d_i}{\sum d_j} \cdot G \tag{5}$$

where $d_i$ is the number of majority class neighbors of $x_i$, and $G$ is the total number of synthetic samples to generate. By dynamically adjusting the distribution of generated samples based on their difficulty level, ADASYN helps models focus on improving the classification of challenging data points. Similar to SMOTE, ADASYN enhances the representation of minority classes without duplication, thereby mitigating the risks of overfitting. However, the computational cost of identifying hard-to-classify samples and generating additional data may increase training time. Furthermore, the method's reliance on minority class density can sometimes result in less representative synthetic samples for sparse datasets.

Resampling techniques like RUS, ROS, SMOTE, and ADASYN provide valuable tools for addressing class imbalance in datasets, each offering unique advantages and limitations. The choice of technique depends on the specific requirements of the task, including computational constraints, dataset size, and the importance of preserving information from the majority and minority classes.

## 3.3 Related Works

Liu and Gao et al. [47] focus on balancing imbalanced network training data using an ADASYN oversampling technique combined with Light Gradient Boosting Machine (LGBM) to classify NSL-KDD and UNSW-NB15 datasets. Their results show that using ADASYN in conjunction with LGBM outperforms other traditional ML algorithms, and its result is also comparable when using the SMOTE sampling technique. Despite gaining good results, they never compare their results with the results of the balancing test data. Sikha Bagui et al. [42] use ANN with different sampling techniques such as upsampling, downsampling, SMOTE, ADASYN, and their combinations to handle imbalanced training data in the KDD99 and UNSW-NB15 datasets. They find that the combination of upsampling and downsampling, as well as the combination of upsampling and SMOTE, performs very well on extremely imbalanced datasets. Tesfahun et al. [48] used feature selection and SMOTE to balance training data without balancing testing data and built a Random Forest(RF) classifier to classify 5 generic classes in NSL-KDD. Their proposed method generates detection rates above 96% in each class. However, since they did not check for duplicated data after selecting features, their result is likely due to overfitting. To mitigate the class imbalance issue, Jiang and Wang et al. reduce noises from major classes using a one-sided selection technique on training data and increase data in minor classes with SMOTE. Then, they extract spatial features with CNN and temporal features with Bidirectional Long Short-Term Memory (LSTM) to achieve F1 scores of 85.14% and 81.25%, respectively, in the NSL-KDD and UNSW-NB15 datasets[49]. Zhang and Huang et al. handle imbalanced data by combining SMOTE and the clustering-based undersampling Gaussian Mixture Model (GMM), achieving detection rates above 96% for multi-class classification on the UNSW-NB15 and CICIDS2017 datasets [50].

One study [41] investigated the impact of undersampling and oversampling methods on the Morris Power Dataset. The study aimed to compare the performance of ML models trained on an imbalanced dataset versus a balanced dataset obtained through resampling techniques. To address the class imbalance, the researchers first applied undersampling and oversampling techniques to balance the datasets. The study then evaluated the impact of dataset imbalance on intrusion detection performance using a CNN-LSTM model. Specifically, the model was trained on three versions

of the Morris Power dataset: the original imbalanced dataset, a balanced dataset created through undersampling, and a balanced dataset generated through oversampling. The results indicated that the CNN-LSTM model achieved a higher F1-score when trained on the resampled datasets compared to the original imbalanced dataset. For instance, training the model on the undersampled Morris Power dataset improved the F1-score by 8.03% on the test set of the original dataset. These findings highlight the importance of striking a balance between malicious and benign traffic samples to optimize the performance of ML models. However, as highlighted by [40], achieving full balance is not necessarily the optimal strategy in all cases. The optimal resampling rate varies depending on the domain, dataset characteristics, and the specific resampling strategy employed.

In another research, Tran et al. [43] investigated the effect of class imbalance on the performance of ML-based network intrusion detection systems using the NSL-KDD dataset. They categorized the dataset into different attack types and applied various sampling strategies to address class imbalance. These strategies included downsampling and upsampling with SMOTE (DUS), downsampling with SMOTE (DS), and other common resampling methods. To evaluate the impact of resampling, they applied multiple ML models to both the imbalanced and balanced datasets. The experimental results demonstrated that the DUS method, which combines downsampling, upsampling, and the Synthetic Minority Oversampling Technique (SMOTE), achieved the highest F1 score among all tested approaches. This indicates that a hybrid resampling strategy can significantly enhance the performance of intrusion detection systems by effectively balancing the dataset. The study concluded that while resampling techniques can improve model performance, the choice of method significantly influences the results, with hybrid approaches, such as DUS, being particularly effective in addressing class imbalance in network intrusion detection.

Also, recent studies have highlighted the growing role of Generative AI in addressing key challenges of IDS, particularly the persistent issue of class imbalance. A systematic literature review by [51] underscores that IDS datasets often contain disproportionately fewer attack samples compared to normal traffic, leading to biased model training and degraded detection of minority attack classes. The paper emphasizes that generative models, such as Generative Adversarial Networks

(GANs) and Variational Autoencoders (VAEs), have been increasingly adopted to synthesize realistic attack instances, thereby enriching the dataset and mitigating skewed distributions. By generating synthetic minority samples, these methods improve the robustness of classifiers and enhance the detection of rare but critical cyber threats. This line of research demonstrates that leveraging synthetic data not only alleviates imbalance but also contributes to more reliable and generalizable IDS models in IoT and broader cybersecurity environments.

While Generative AI provides promising solutions for mitigating class imbalance in IDS through synthetic data generation, recent work has also drawn attention to its inherent challenges. As noted in [52], generative models may inadvertently introduce noisy or low-quality samples, which can degrade detection performance instead of enhancing it. Moreover, issues such as mode collapse, lack of diversity in generated attacks, and the potential for generating unrealistic traffic patterns pose significant barriers to achieving consistent improvements. The paper further emphasizes the importance of carefully validating synthetic data and designing robust evaluation strategies to avoid overfitting to artificial patterns. These concerns highlight that although generative approaches can alleviate imbalance, their practical deployment in IDS requires addressing critical quality, reliability, and generalization limitations.

Building on these insights, [53] introduce SYN-GAN, a GAN-driven NIDS pipeline that explicitly targets class imbalance by generating class-conditional synthetic traffic to equalize both the normal/attack proportions and the per-attack distributions across benchmark datasets (UNSW-NB15, NSL-KDD, BoT-IoT), followed by training conventional classifiers on the synthetic set and validating on real data. Notably for imbalance remediation, the authors highlight prior evidence that pairing GAN-based augmentation with an RF classifier yields more effective results than SMOTE for class balancing on CICIDS2017—underscoring that GenAI-driven sampling can outperform classical oversampling when coupled with robust tree ensembles. Empirically, their synthetic-only training regime attains strong real-data performance, supporting the use of high-quality synthetic minorities to improve detection of rare classes without extensive real-data collection; together, these findings motivate adopting GAN-based, distribution-aware resampling over SMOTE when addressing severe skew in IoT-IDS corpora.

Recent work by Fairstein et al. [54] investigates the challenge of class imbalance in active learning for imbalanced datasets, a problem that is also critical in IDS. While traditional active learning strategies focus on sample diversity and model uncertainty, they often fail to adequately capture minority class instances, which are essential for robust detection performance. To address this, the authors introduce a novel self-tuned weighting method (InvProp) that implicitly balances the dataset by skewing the selection process toward informative minority examples without requiring manual tuning. Their experiments on multiple NLP benchmarks show that this strategy significantly increases the number of minority samples acquired and improves balanced accuracy and ROC-AUC compared to standard baselines. Importantly, they also highlight that class imbalance can be mitigated by either generating additional minority samples or reweighting strategies that approximate synthetic balancing, both of which have implications for applying generative AI to IDS, where synthetic data generation can alleviate skewed distributions and enhance classifier robustness.

In summary, the reviewed studies as shown in Table 3.3 emphasize the effectiveness of resampling techniques—such as SMOTE, ADASYN, and hybrid approaches—in improving model performance on imbalanced cybersecurity datasets. However, a common limitation across these works is their focus solely on balancing the training data, while leaving the testing data in its original imbalanced form. Although this approach helps assess generalizability under real-world conditions, it may also obscure the full potential or drawbacks of different resampling strategies when applied more broadly.

This observation reveals a clear research gap: the impact of resampling both training and testing datasets has not been systematically investigated. To address this, our study aims to evaluate and compare the performance of various resampling methods under two distinct scenarios: when only the training data is balanced and when both the training and testing datasets are resampled, using one IT (CICIDS2017) and one OT (SWaT) dataset. This investigation offers new insights into how resampling strategies and scopes impact model performance, particularly under varying data characteristics across IT and OT environments.

Table 3.3: Summary of Existing Works on Handling Imbalanced Datasets in Intrusion Detection

| **Authors** | **Techniques Used** | **Datasets and Models** | **Testing Data Balanced** |
|---|---|---|---|
| Liu and Gao et al. [47] | ADASYN oversampling with LGBM | NSL-KDD, UNSW-NB15; Compared with SMOTE | **No** |
| Sikha Bagui et al. [42] | Upsampling, Downsampling, SMOTE, ADASYN, combinations | KDD99, UNSW-NB15; Used ANN | **No** |
| Tesfahun et al. [48] | Feature Selection + SMOTE on training data | NSL-KDD; Used Random Forest | **No** |
| Jiang and Wang et al. [49] | One-side selection + SMOTE; CNN + BiLSTM | NSL-KDD, UNSW-NB15 | **No** |
| Zhang and Huang et al. [50] | SMOTE + Clustering-based undersampling (GMM) | UNSW-NB15, CICIDS2017 | **No** |
| Balla et al. [41] | Undersampling, Oversampling; Compared original vs. balanced data | Morris Power; Used CNN-LSTM | **No** |
| Tran et al. [43] | DS, DUS, SMOTE; Hybrid resampling | NSL-KDD; Multiple ML models | **No** |

## 3.4 Research Questions

Based on the further review of existing methods, we have identified the following research questions for this chapter:

### 3.4.1 How to determine the best resampling technique for a specific dataset?

One of the major challenges in designing intrusion detection systems for ICS/OT environments is the pervasive issue of class imbalance in datasets. When datasets are dominated by normal traffic, ML models often become biased toward the majority class, leading to misclassification of the critical minority events. Various resampling techniques—such as ROS, RUS, SMOTE,

and ADASYN—have been developed to address this imbalance. However, no single method has emerged as universally optimal across different datasets, since the effectiveness of each technique is highly dependent on the specific characteristics of the dataset, including the degree of imbalance, the diversity of attack types, and the underlying data distribution.

Determining the best resampling technique for a given dataset is essential because an inappropriate method may either fail to adequately represent the minority class or inadvertently introduce issues such as overfitting or loss of critical information. The optimal choice should be driven by the specific requirements of the dataset and the target application, ensuring that the model effectively learns the characteristics of both benign and malicious traffic. A systematic evaluation of various resampling approaches can yield insights into their impact on key performance metrics—such as precision, recall, and F1 score—which are crucial for reliable intrusion detection. This evaluation is fundamental to designing models that are robust and generalize well to real-world conditions.

### 3.4.2 What is the impact of balancing train and test sets?

s far as imbalanced classification problems are concerned, researchers have traditionally focused on balancing the training set to enhance the model's ability to learn from minority class examples. However, a less commonly addressed yet critical question involves the impact of balancing the test set in addition to the training set. This question becomes especially important when evaluating models in scenarios where the distribution of the test data does not reflect the real-world class imbalance. In many real-life intrusion detection systems, malicious traffic represents only a small fraction of the total traffic, leading to naturally imbalanced environments. Evaluating models on artificially balanced test sets in such cases may produce overly optimistic results and fail to reveal how the model would perform in actual deployment.

To answer this, experiments are designed using different sampling techniques under various configurations: (1) both training and testing sets imbalanced, (2) only the training set balanced, and (3) both sets balanced. The purpose is to identify which setup yields the most reliable and realistic evaluation.

## 3.5 Methodology

This section outlines the steps required to address the research questions and achieve the goal of this chapter, which is to determine the appropriate resampling technique for the selected dataset.

### 3.5.1 Dataset Selection

For this experiment, we chose the CIC-IDS 2017 dataset [28] as an IT dataset and the SWaT dataset as an OT dataset. The CICIDS2017 dataset contains 2,830,473 network traffic samples, with benign traffic accounting for 80.30% and attack traffic accounting for 19.70 %. The categories include the most prevalent attacks, such as DoS, DDoS, Botnet, PortScan, web attacks, etc. The dataset comprises 84 features from the generated network traffic, with the multiclass label appearing in the last column. Modern industrial environments increasingly integrate IT and OT systems, resulting in converged networks where IP-based communication is prevalent. This convergence means that many attack vectors, such as DoS attacks, infiltration, and data exfiltration—observed in CIC-IDS2017 are also relevant to OT environments. Moreover, although CIC-IDS2017 does not include OT-specific protocols like Modbus or DNP3, the underlying network behavior and traffic patterns can be similar to those in OT networks that rely on standard TCP/IP communications. As a result, techniques and insights gained from experiments on CIC-IDS2017, particularly those related to resampling methods for handling imbalanced data, may provide valuable guidance for developing and fine-tuning intrusion detection systems in OT settings.

On the other hand, the Secure Water Treatment (SWaT) dataset is a widely used OT dataset collected from a real-world industrial water treatment testbed. It captures various cyberattack scenarios in a multi-stage water purification process, including physical and network-based attacks. The dataset is highly imbalanced, as attacks occur infrequently compared to normal operations, making it an ideal candidate for evaluating resampling methods. Implementing CNN-LSTM on the SWaT dataset can effectively capture both spatial and temporal dependencies in ICS data, improving the detection of rare but critical cyberattacks. By applying different resampling techniques such as ROS, RUS, SMOTE, ADASYN, and one-sided selection, the impact of class balancing on model performance can be assessed, helping to determine the most effective resampling strategy for

real-world ICS intrusion detection.

### 3.5.2 Machine Learning Model Description

The CNN-LSTM method combines the strengths of both CNNs and LSTM networks, making it particularly effective for intrusion detection in imbalanced datasets. CNNs efficiently extract spatial features from network traffic data, capturing key attack patterns, while LSTMs process sequential dependencies, enabling the model to recognize temporal attack behaviors. This hybrid approach outperforms traditional ML methods by leveraging both spatial and temporal information, making it robust against complex attack patterns. Additionally, CNN-LSTM is more resilient to overfitting than standalone deep learning models when applied to resampled datasets, as it generalizes better across varying class distributions. This specific fusion has garnered significant popularity among researchers due to its enhanced effectiveness in handling complex time-series data and improving classification accuracy, particularly in cases of imbalanced datasets when combined with appropriate resampling techniques. As such, CNN-LSTM becomes a preferred choice in the field of cybersecurity research as it effectively learns both spatial and temporal dependencies in attack patterns, thus providing a more robust and adaptive system for intrusion detection.

We employed a hybrid CNN-LSTM model for binary classification on the CICIDS2017 and SWaT datasets. The model architecture begins with a 1D convolutional layer containing 64 filters and a kernel size of 1, activated by the ReLU function to extract low-level spatial features. This is followed by an LSTM layer with 64 memory units to capture temporal dependencies across time steps. The output is then flattened and passed through a dense layer with 64 units and ReLU activation, culminating in a softmax-activated output layer to predict multi-class labels. We used the Adam optimizer and categorical cross-entropy loss function, which is suitable for multi-class classification tasks. The model was trained with a validation split of 20% from the training set. Before training, all features were normalized using MinMaxScaler to bring them within the [0, 1] range, and the input was reshaped into a 3D format with a time step size of 10 to suit the CNN-LSTM structure.

### 3.5.3 Experiment Description

Eight experiments were conducted to determine the effect of dataset imbalance using the CNN-LSTM method. In the first experiment, CNN-LSTM detected intrusions using imbalanced data. As Figure 3.2 illustrates the flowchart of balancing the datasets and training the model, in the second phase, the data were balanced in two different scenarios using resampling methods. In the first scenario, only the training data becomes balanced, and the testing data remains imbalanced. The second scenario is implemented by balancing both training and testing data to compare the results. The model is trained with balanced data methods such as undersampling, oversampling, and hybrid balancing. Next, the CNN-LSTM model is used to detect anomalies in the dataset. The SCADA datasets are adjusted using the MinMaxScaler with 80% of the data used for training and 20% for testing. In experiments 2, 3, and 4, the dataset was divided based on its majority and minority classes. To balance the dataset, the majority class was undersampled, and the minority class was oversampled [41].

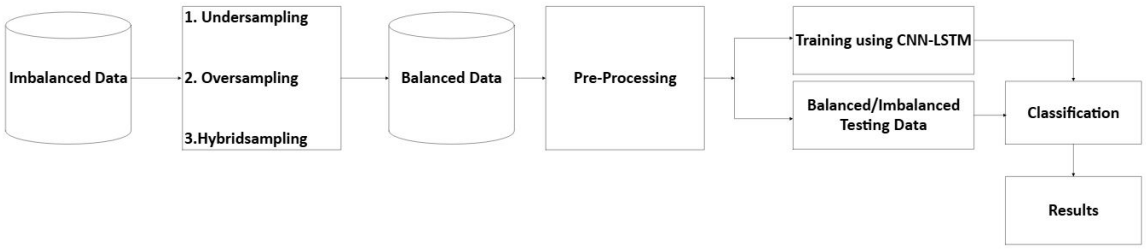


Figure 3.2: The flowchart of balancing the datasets and training the CNN-LSTM model with balanced data.

### 3.5.4 Performance Evaluation

In this part, we present the evaluation metrics. The evaluation metrics presented below are defined in terms of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The metrics used for this purpose are as follows [55]:

**Accuracy**

It is a metric that generates the number of correct predictions over the total number of predictions made by the model. Accuracy is best suited for a balanced dataset and biased for an imbalanced dataset.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6}$$

**Precision**

It is a metric that generates correct optimistic predictions, which are divided by the total number of positive predicted values. A low precision value indicates a high number of False Positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

**Recall**

This metric is generated by the True Positives' ratio to the sum of TPs and FNs. In this case, a low recall value indicates a high number of False Negatives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

**F1-Score**

It is the harmonic mean value generated between recall and precision measures.

$$\text{F1-Score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{9}$$

## 3.6 Results

In this section, we present the numerical results of our experiments in detecting attacks through different dataset-balancing approaches, as mentioned in the previous subsection. The primary objective of this section is to provide answers to the research questions. We discussed the details, including the evaluation metrics used and the outcome, using the CNN-LSTM model to classify

attacks in SCADA network data to answer the research questions [41].

### 3.6.1 Results of Imbalanced Dataset

Initially, we compared dataset performance without employing any balancing strategy. The datasets were divided into two groups for binary classification: benign and attack, as illustrated in Table 3.4. Table 3.5 presents the exact values for the evaluation metrics, which are the accuracy, precision, recall, and F1-score.

Table 3.4: The Binary Classification With CNN-LSTM

| Dataset | No. of Records | Type of Records | No. of Classes |
|---|---|---|---|
| CICIDS 2027 | 2,830,743 | Normal And Attacks | 2 |
| SWaT | 449,919 | Normal And Attacks | 2 |

Table 3.5: Performance results For The Imbalanced Datasets

| Dataset | ACC | Precision | Recall | F1-Score |
|---|---|---|---|---|
| CIC-IDS2017 | 98.63 | 96.67 | 96.04 | 96.34 |
| SWaT | 97.36 | 92.46 | 86.84 | 89.40 |

### 3.6.2 Results of the Balanced Dataset

#### Undersampling

The second phase of our experiment involved sampling the data using undersampling techniques to initially balance the dataset. The datasets were balanced in two different situations. The first experiment involves sampling only the training data, while the test data remains imbalanced. The second experiment aims to balance both training and testing data using undersampling algorithms, such as RUS, one-sided selection, and nearmiss algorithms. The data were then divided into training and testing segments with an 80:20 split. The CNN-LSTM model was trained with the balanced dataset.

Tables 3.6 and 3.7 display the results of the application of the undersampling method to balance the CICIDS 2017 dataset for both testing and training. In terms of a balanced data set, the near-miss algorithm delivered the best results. The model's performance increased by 2.7% to a maximum

of 98.89% F1-Score. The random undersampler achieved 98.50% F1-Score while the one-sided selection generated an F1-Score of 97.95%. Overall, the performance of the CICIDS2017 dataset is excellent. The results for imbalanced testing data, as shown in Table 3.8, are slightly lower than the balanced testing results.

Table 3.6: Balancing CICIDS2017 Dataset With Undersampling

| **Technique** | **Before** | | **After** | |
|---|---|---|---|---|
| | **Normal** | **Attack** | **Normal** | **Attack** |
| Random Under Sampling | 2,273,097 | 557,646 | 1,101,448 | 557,646 |
| One Sided Selection | 2,273,097 | 557,646 | 1,897,651 | 557,646 |
| Near Miss | 2,273,097 | 557,646 | 1,291,001 | 557,646 |

Table 3.7: Performance Results For The Balanced Testing CICIDS2017 Dataset With Undersampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Random | 98.38 | 98.81 | 98.19 | 98.50 |
| One Sided Selection | **99.97** | 97.62 | 98.29 | 97.95 |
| **Near Miss** | 99.75 | 99.33 | 98.87 | **98.89** |

Table 3.8: Performance Results For The Imbalanced Testing CICIDS2017 Dataset With Undersampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Random | 97.23 | 97.29 | 97.23 | 97.22 |
| One Sided Selection | **99.96** | 96.75 | 98.11 | 97.44 |
| **Near Miss** | 98.48 | 98.80 | 98.97 | **98.88** |

The results of balancing the both training and testing data of SWaT dataset are shown in Table 3.10. The near-miss method produced the best results. Table 3.9 shows the binary classification results of the balanced SWaT dataset. The near-miss undersampler improves the F1-score significantly than the other algorithms, which is the primary metric in our research. The performance improved by 9% compared to the unbalanced dataset. The unbalanced dataset has an F1-score of 89.40% while the balanced dataset has an F1-score of about 98%.

On the other hand, the results of imbalanced testing data are shown in Table 3.11. The NearMiss method results have dropped significantly from 95.89% of F1-Score when the testing data was

balanced to 85.17%.

Table 3.9: Balancing SWaT Dataset With Undersampling.

| **Technique** | **Before** | | **After** | |
|---|---|---|---|---|
| | **Normal** | **Attack** | **Normal** | **Attack** |
| Random | 395,298 | 54,621 | 109,242 | 54,621 |
| One Sided Selection | 395,298 | 54,621 | 281,268 | 54,621 |
| Near Miss | 395,298 | 54,621 | 114,719 | 54,621 |

Table 3.10: Performance Results For The SWaT Balanced Testing Data With Undersampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Random | 96.99 | 90.43 | 95.61 | 92.94 |
| One Sided Selection | **98.54** | 91.68 | 96.20 | 93.89 |
| **Near Miss** | 96.23 | 96.29 | 95.49 | **95.89** |

Table 3.11: Performance results for the SWaT imbalanced testing data with undersampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| **Random** | **97.68** | 97.74 | 98.33 | **97.70** |
| One Sided Selection | 95.71 | 90.21 | 96.22 | 93.08 |
| Near Miss | 82.63 | 91.73 | 82.63 | 85.17 |

### Oversampling

This section describes the third experiment, which only used oversampling approaches to balance the datasets. ROS, SMOTE, and ADASYN were the algorithms used to balance the data.

The SMOTE algorithm outperformed ROS and ADASYN in oversampling the minority class. Table 3.13 displays the outcome of the CNN-LSTM binary classification with the oversampled CICIDS dataset. The accuracy increased from 98.63% when SMOTE was used to 99.66%. Also, when ADASYN was used, the F1-score dropped from 96.34% to 94.81%. Table 3.14 provides the imbalanced testing data results of CICIDS2017 with oversampling methods. As we can see, SMOTE gained the best results with 99.00% F1-Score.

Table 3.12: Balancing The CICIDS2017 Dataset With Oversampling.

| **Technique** | **Before** | | **After** | |
|---|---|---|---|---|
| | **Normal** | **Attack** | **Normal** | **Attack** |
| Random | 2,273,097 | 557,646 | 2,273,097 | 1,136,422 |
| SMOTE | 2,273,097 | 557,646 | 2,273,097 | 1,275,293 |
| ADASYN | 2,273,097 | 557,646 | 2,273,097 | 988,426 |

Table 3.13: Performance Results For The Balanced CICIDS2017 Dataset With Oversampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Random | 99.00 | 98.24 | 98.26 | 98.25 |
| **SMOTE** | **99.66** | 98.28 | 99.72 | **99.00** |
| ADASYN | 97.89 | 97.49 | 92.27 | 94.81 |

Table 3.14: Performance Results For The Imbalanced CICIDS2017 Dataset With Oversampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Random | 98.45 | 98.50 | 98.45 | 98.46 |
| **SMOTE** | 98.45 | 99.51 | 98.66 | **99.00** |
| ADASYN | **98.52** | 98.54 | 95.28 | 96.88 |

Table 3.15 displays the class distribution for the SWaT dataset. SMOTE oversamples the normal class, resulting in a dataset that is nearly balanced. ADASYN performed well when oversampling the minority category, but did not perform optimally. In terms of performance, the accuracy of all algorithms is around 98%. The difference is evident in the other metrics; for example, the SMOTE algorithm performed best in the F1-score, scoring 96.50%; the detailed findings are shown in Table 3.16.

Also, Table 3.17 shows the results when the testing data is imbalanced. The results for SMOTE and ADASYN are slightly lower than when the testing data is balanced; However, the F1-Score increased from 93.65 to 94.04 for random oversampling.

Table 3.15: Balancing The SWaT Dataset With Oversampling.

| Technique | Before | | After | |
|---|---|---|---|---|
| | **Normal** | **Attack** | **Normal** | **Attack** |
| Random | 395,298 | 54,621 | 395,298 | 197,649 |
| SMOTE | 395,298 | 54,621 | 395,298 | 235,773 |
| ADASYN | 395,298 | 54,621 | 395,298 | 189,583 |

Table 3.16: Performance Results For The SWaT Balanced Testing Data With Oversampling.

| Technique | ACC | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Random | 98.25 | 91.42 | 96.00 | 93.65 |
| **SMOTE** | **98.63** | 95.85 | 97.23 | **96.50** |
| ADASYN | 97.47 | 91.60 | 95.47 | 93.54 |

Table 3.17: Performance Results For The SWaT Imbalanced Testing Data With Oversampling.

| Technique | ACC | Precision | Recall | F1-Score |
|---|---|---|---|---|
| **Random** | 95.78 | 92.57 | 95.59 | **94.04** |
| SMOTE | **98.12** | 90.41 | 96.33 | 93.26 |
| ADASYN | 97.50 | 92.03 | 90.34 | 91.24 |

### Hybrid Sampling

The dataset was balanced in the fourth and final experiment using a combination of under-sampling and oversampling methods. The class distribution of balancing the CICIDS2017 dataset using a hybrid technique is shown in Table 3.18. The performance of the model decreased. When ADASYN was combined with the near-miss algorithm for hybrid balancing, a roughly similar result was obtained. The detailed values for the evaluation metrics are provided in Table 3.19. In the CICIDS 2017 dataset, the accuracy significantly decreased from 96.34% to 86.06% when SMOTE and NearMiss were combined. According to Table 3.20, ADASYN with NearMiss gained the highest

results just like when the testing data is balanced.

Table 3.18: Balancing The CICIDS2017 Dataset With Hybrid Sampling.

| **Technique** | **Before** | | **After** | |
|---|---|---|---|---|
| | **Normal** | **Attack** | **Normal** | **Attack** |
| SMOTE and Near Miss | 2,273,097 | 557,646 | 1,768,395 | 801,467 |
| ADASYN and Near Miss | 2,273,097 | 557,646 | 1,384,722 | 978,504 |

Table 3.19: Performance Results For The Balanced Testing CICIDS2017 Dataset With Hybrid Sampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| SMOTE and Near Miss | 86.06 | 87.66 | 81.00 | 84.17 |
| **ADASYN and Near Miss** | **89.63** | 85.67 | 86.06 | **85.84** |

Table 3.20: Performance Results For Imbalanced Testing CICIDS2017 Dataset With Hybrid Sampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| SMOTE and Near Miss | 84.56 | 85.09 | 83.72 | 84.55 |
| **ADASYN and Near Miss** | **87.41** | 85.32 | 86.16 | **85.64** |

As shown in Table 3.21, for the balanced SWaT dataset, the first coupled algorithm was SMOTE and near miss. This method successfully balanced the SWaT dataset. The detailed values for the evaluation metrics are provided in Table 3.22. In the SWaT dataset, accuracy decreased significantly from 98% to 90%.

According to Table 3.23, which shows the results of hybrid sampling while the testing data is imbalanced, ADASYN + NearMiss method gained better results than SMOTE + NearMiss with 87.24% F1-Score; however, the other method has better accuracy.

Table 3.21: Balancing The SWaT Dataset With Hybrid Sampling.

| **Technique** | **Before** | | **After** | |
|---|---|---|---|---|
| | **Normal** | **Attack** | **Normal** | **Attack** |
| SMOTE and Near Miss | 395,298 | 54,621 | 282,355 | 197,649 |
| ADASYN and Near Miss | 395,298 | 54,621 | 262,118 | 197,483 |

Table 3.22: Performance Results For The SWaT Balanced Testing Data With Hybrid Sampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| SMOTE and Near Miss | **90.84** | 86.00 | 87.44 | 86.76 |
| **ADASYN and Near Miss** | 89.88 | 87.56 | 88.61 | **88.05** |

Table 3.23: Performance Results For The SWaT Imbalanced Testing Data With Hybrid Sampling.

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| SMOTE and Near Miss | **91.06** | 84.28 | 90.05 | 87.06 |
| **ADASYN and Near Miss** | 87.47 | 88.18 | 86.32 | **87.24** |

## 3.7 Discussions

In the previous section, we attempted to find out how dataset imbalances affected IDSs. To fully understand this effect, we conducted four experiments, balancing IT and OT imbalanced datasets by employing undersampling, oversampling, or a combination of both methods. Overall results of the CICIDS2017 dataset are shown in Table 3.24. Despite the SMOTE technique, a very popular oversampling method, achieving an individual F1-score of 99%, the overall performance of the oversampling category lagged behind that of undersampling. This decrease is primarily due to the poor performance of ADASYN, another oversampling technique that failed to improve the assessment measures significantly. This resulted in a decrease in the overall mean performance measures of the oversampling category, despite SMOTE's strong individual performance. On the other hand, undersampling-related methods such as Random UnderSampling (RUS), One-Sided Selection, and

NearMiss showed high and consistent performance. Most prominently, the maximum F1-score of 99.32% was achieved by NearMiss in the range of undersampling methods. As all the undersampling methods demonstrated good performance, the overall mean performance of these methods was high, indicating high stability in several measures, as evidenced by the low standard deviation. However, when the dataset was balanced with hybrid sampling, the evaluation metrics dropped significantly. This is due to the dataset being distorted by the process of adding and removing records. Finally, we can conclude that oversampling (SMOTE) yields the best results for the CICIDS-2017 dataset when implemented with the CNN-LSTM method.

Table 3.24: Overall Results For The Balanced CICIDS2017 Testing Data

| **Technique** | **ACC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Imbalanced | 98.63 | 96.67 | 96.04 | 96.34 |
| **Undersampling** | **99.39 +/- 0.86** | **98.73 +/- 1.07** | **98.45 +/- 0.36** | **98.59 +/- 0.68** |
| Oversampling | 98.85 +/- 0.89 | 97.47 +/- 0.60 | 96.75 +/- 3.94 | 97.35 +/- 2.23 |
| Hybrid Sampling | 87.85 +/- 2.52 | 86.66 +/- 1.41 | 83.53 +/- 3.58 | 84.51 +/- 1.37 |

The results in Table 3.25 demonstrate that for the CICIDS2017 dataset under imbalanced testing conditions, oversampling yields the best overall performance, achieving the highest average F1-score of 98.11% with very low variance across metrics. This suggests that synthetic data generation techniques such as SMOTE effectively enrich the minority class without discarding useful majority class data, leading to a more balanced learning process and robust generalization. Undersampling also performs well, with an F1-score of 97.85%, but exhibits slightly higher variance, likely due to the reduced size of the training data, which may lead to occasional underfitting or sensitivity to sample selection. The imbalanced baseline, while achieving a high accuracy of 98.63%, has a lower F1-score (96.34%), reflecting the model's tendency to favor the majority class, thus missing minority class instances. Hybrid sampling results in a substantial performance drop (84.10% F1-score), indicating that simultaneous under- and oversampling may introduce inconsistencies or redundant patterns, harming the model's ability to distinguish between classes in the presence of skewed test data. Overall, oversampling stands out as the most effective strategy when evaluated on the imbalanced CICIDS2017 test set.

The overall results of the SWaT dataset are shown in Table 3.26. It is clear from the pattern

Table 3.25: Overall Results For The CICIDS2017 Imbalanced Testing Data

| Technique | ACC | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Imbalanced | 98.63 | 96.67 | 96.04 | 96.34 |
| Undersampling | 98.56 +/- 1.37 | 97.61 +/- 1.28 | 98.10 +/- 0.87 | 97.85 +/- 0.90 |
| **Oversampling** | **98.47 +/- 0.04** | **98.52 +/- 0.02** | **97.13 +/- 1.64** | **98.11 +/- 0.90** |
| Hybrid Sampling | 85.99 +/- 2.47 | 85.21 +/- 0.16 | 84.94 +/- 1.73 | 84.10 +/- 0.78 |

that when using the SWaT dataset, the CNN-LSTM model performs quite modestly. This is due to the dataset's small size, which is only around 450,000. The outcome may differ if a ML algorithm is used. Deep learning techniques, however, need a bigger dataset. As shown here, oversampling achieves the highest F1-score of 94.56%.

Table 3.26: Overall Results For The SWaT Balanced Testing Data

| Technique | ACC | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Imbalanced | 97.36 | 92.46 | 86.84 | 89.40 |
| Undersampling | 97.25 +/- 1.17 | 92.80 +/- 3.08 | 95.77 +/- 0.38 | 94.24 +/- 1.50 |
| **Oversampling** | **98.12 +/- 0.59** | **92.96 +/- 2.51** | **96.23 +/- 0.90** | **94.56 +/- 1.68** |
| Hybrid Sampling | 90.36 +/- 0.67 | 86.78 +/ 1.10 | 88.02 +/- 0.82 | 89.40 +/- 0.91 |

The results in Table 3.27 demonstrate that for the SWaT dataset under imbalanced testing conditions, oversampling yields the best overall performance, achieving the highest average F1-score of 92.85% with low variability across all metrics. This suggests that generating synthetic samples using methods like SMOTE helps the model learn a more balanced representation of the minority class without losing valuable data from the majority class, crucial in time-series industrial datasets like SWaT, where temporal patterns are important. Undersampling, while achieving decent performance (91.98% F1-score), suffers from high variance (±6.33), indicating instability that may be due to the loss of useful majority class data, which negatively impacts generalization. The hybrid approach performed the worst, with an F1-score of 87.12%, likely because the combination of adding and removing samples disrupts the temporal integrity and correlations within the SWaT sequences, which are essential for detecting cyber-physical anomalies. Interestingly, even the imbalanced baseline model performs reasonably well (89.40% F1), highlighting that deep learning models like CNN-LSTM can still manage some imbalance; however, their performance improves with informed balancing—particularly oversampling in this context.

Table 3.27: Overall Results For The SWaT Imbalanced Testing Data

| Technique | ACC | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Imbalanced | 97.36 | 92.46 | 86.84 | 89.40 |
| Undersampling | 92.01 +/- 8.18 | 93.23 +/- 3.98 | 92.39 +/- 8.52 | 91.98 +/- 6.33 |
| **Oversampling** | **97.13 +/- 1.21** | **91.67 +/- 1.12** | **94.09 +/- 3.27** | **92.85 +/- 1.45** |
| Hybrid Sampling | 89.27 +/- 2.54 | 86.23 +/ 2.76 | 88.18 +/- 1.86 | 87.12 +/- 0.09 |

## 3.8 Recommendations

In this section, we answer the questions that have been asked in the problem formulation.

Firstly, we evaluated how existing resampling techniques affect the performance of IDSs on both IT (CICIDS2017) and OT (SWaT) datasets under the CNN-LSTM deep learning architecture. Our results demonstrate that balancing the dataset—particularly through oversampling techniques such as SMOTE—can significantly enhance the detection performance.

Based on our experimental results, we recommend a cautious choice of sampling approach that considers both the size of the dataset and the balance of test data. For large datasets like CICIDS2017, undersampling showed remarkable efficacy when test data were imbalanced, achieving robust recall and F1-score with acceptable stability. This indicates that removing duplicate majority samples can result in reduced training time with maintained model performance, especially when sufficient data exist for all classes. On the other hand, for small datasets like SWaT, oversampling methods, such as SMOTE, consistently outperformed other methods, especially in imbalanced test conditions. This may be due to the loss of important temporal or contextual data that accompanies the undersampling of small datasets. Additionally, when both test and training data are balanced—an ideal but unrealistic situation—oversampling typically performs better in both datasets for all measures, indicating its ability to generate informative synthetic examples without compromising the integrity of the original samples. For these reasons, we suggest deploying oversampling for small datasets and undersampling for large datasets, especially when test data are imbalanced, to achieve an effective balance between performance and applicability.

### 3.8.1 How to properly synthesize a dataset for ML-based IDS?

To achieve realistic yet effective synthetic datasets, the imbalance ratio should be tailored to the specific use case. For general intrusion detection tasks, a moderate imbalance ratio of 80:20 or 90:10 often provides the best trade-off between model performance and real-world applicability. This ratio ensures sufficient representation of the attack without overwhelming the dataset with minority-class samples. However, for rare, high-impact attacks, synthetic datasets should prioritize these classes by adjusting the imbalance ratio to favor their inclusion, ensuring the model learns to detect critical threats effectively.

Moreover, dataset synthesis should consider the diversity of attack scenarios. Including various attack types, such as DoS, MITM, and spoofing, further enhances dataset utility. By carefully balancing synthetic datasets based on the intended use case and attack diversity, researchers can develop more robust and reliable intrusion detection systems for ICS environments.

### 3.8.2 Different Resampling Strategies for Different Use Cases and Attack Classes

Selecting the appropriate resampling technique for addressing class imbalance in ICS cybersecurity datasets depends on the specific use case, the type of attack, and the characteristics of the dataset. Different attack classes exhibit varying patterns of imbalance, requiring tailored resampling approaches to achieve optimal ML performance.

For use cases involving rare, high-impact attacks such as Stuxnet-like or zero-day attacks, oversampling techniques like the SMOTE or ADASYN are more effective. These methods generate synthetic samples to enrich the minority class without simple duplication, enhancing the model's ability to identify rare attack patterns. ADASYN, in particular, focuses on hard-to-classify instances, making it suitable for datasets where minority-class samples are scattered across feature space.

In contrast, for more frequent attack types, such as DoS or port-scanning attacks, RUS may be preferable, as it reduces the volume of majority-class data while retaining representative samples. This method helps maintain computational efficiency and prevents the model from being overwhelmed by redundant normal traffic.

Hybrid approaches, such as Downsampling + Upsampling + SMOTE (DUS), are particularly effective for datasets exhibiting multiple imbalance levels across different attack categories. These methods combine the strengths of both oversampling and undersampling, ensuring a balanced distribution while preserving data diversity. For real-time intrusion detection in ICS environments, dynamic resampling strategies that adjust the imbalance ratio based on data stream characteristics can further enhance detection accuracy while minimizing latency.

## 3.9 Conclusion

This chapter provided a comprehensive analysis of resampling strategies to address class imbalance in cybersecurity datasets, with a particular focus on ML-based intrusion detection for both IT (e.g., CICIDS2017) and OT (e.g., SWaT) environments. A literature review revealed a significant gap in systematically selecting appropriate resampling methods tailored to the dataset type (large vs. small) and the balancing scope (only training vs. both training and testing). To address this, multiple oversampling, undersampling, and hybrid sampling techniques were evaluated under two experimental scenarios: (1) balancing only the training data and (2) balancing both training and testing data. The results demonstrated that for large datasets like CICIDS2017, oversampling the training data alone using methods such as SMOTE is typically sufficient, as it preserves the integrity of the test set for realistic performance evaluation; however, undersampling both the training and testing data for large datasets may lead to having better results. In contrast, small datasets like SWaT may benefit from balancing both training and testing data by oversampling techniques to ensure adequate representation of minority attack classes, particularly rare and critical ones. Ultimately, the choice of resampling method should be informed by the severity of imbalance, dataset size, the importance of attack detection, and computational constraints. By addressing the current gap in adaptive resampling selection, this work offers practical guidance for building more accurate and resilient intrusion detection systems.

# Chapter 4

# Integration of Time-To-Detection As a Temporal Performance Metric

## 4.1 Introduction

Detection time is a crucial feature for IDS in real-time networks of critical applications, where even slight delays can lead to significant disruptions or failures. Time-To-Detection (TTD) is a critical metric in cybersecurity for smart grid applications [56]. In OT environments, the primary security objective is availability, followed by integrity and confidentiality. Therefore, TTD becomes a vital metric in OT cybersecurity, as delayed detection can result in significant operational failures or even physical harm. TTD representing the total time required to detect intrusion data and classify it as a cyberattack or normal. The ability to detect attacks promptly ensures that mitigation strategies can be applied before significant disruptions occur.

Unlike conventional evaluation metrics such as accuracy, precision, or F1-score, which primarily assess the correctness of detection, the TTD metric emphasizes the temporal efficiency of detection. In time-sensitive industrial environments, particularly within OT networks, even high-accuracy models become ineffective if detection occurs after the appropriate response window has expired. By focusing the evaluation on TTD, this research prioritizes not only whether an attack is detected, but also when it is detected, aligning the analysis with the operational realities and safety requirements of ICS infrastructures.

## 4.2 Related Work

In this study, a thorough review of the literature on TTD in IDS reveals significant inconsistencies in how TTD is defined and measured across previous research. The TTD in existing literature is typically measured as the time between the start of the attack and when the attack is detected. This general definition does not involve all related procedures, such as log transmission, collection, analysis, and subsequent alert generation.

Zhou et al.[50] introduced a novel method called Deep Feature Embedding Learning (DFEL), which leverages deep learning and transfer learning to enhance the efficiency and effectiveness of cyberattack detection systems. DFEL aims to reduce data dimensionality while preserving essential information by using deep neural networks to extract high-level features from raw input data. These features are then used to accelerate the performance of traditional ML classifiers. The authors applied DFEL to the NSL-KDD and UNSW-NB15 datasets, demonstrating its ability to reduce training and prediction times across multiple classifiers significantly. Notably, the method improved the recall of the Gaussian Naive Bayes classifier from 80.74% to 98.79% and reduced the SVM's prediction time from 67.26 seconds to just 6.3 seconds. These results highlight DFEL's potential to substantially reduce time-to-detection while improving detection accuracy, making it a valuable contribution to real-time intrusion detection systems. However, despite focusing on speed, the paper does not provide a complete definition or measurement of TTD as proposed in our model. Specifically, their evaluation of "detection time" focuses solely on model inference speed and classifier runtime, without considering other crucial components such as the delay between the actual start of an attack and the receipt of the log by the IDS or the alert generation delay after detection.

In another research, Ozalp and Albayrak [57] investigated the impact of high-frequency feature selection on both detection accuracy and runtime in NSL-KDD–based IDS. They applied six attribute-selection methods (OneR, Chi-Square, CBS, SUC, GR, IG) to identify the ten most frequent features, then evaluated Random Forest (RF), J48, Naive Bayes (NB), and Multi-Layer Perceptron (MLP) classifiers on this reduced set. Their results showed that RF and J48 achieved the lowest detection times—around 6.78 s—while maintaining high accuracy and precision. This work

demonstrates how feature selection can substantially reduce model processing time, one key component of TTD. However, their evaluation focuses solely on classifier runtime and does not account for other critical delay components in an operational ICS setting. Specifically, they omit the delay from actual attack initiation to log collection and the latency incurred in raising an alert after detection. Without these stages, their reported "detection time" underestimates the end-to-end delay that would be experienced in practice.

One notable contribution to adaptive detection timing in Cyber-Physical Systems (CPS) is presented in Akowuah et al. [58], where the authors propose a real-time sensor attack detection framework that explicitly balances detection delay and false alarm rate through awareness of the system state. Recognizing that minimizing both metrics simultaneously is often infeasible, the framework introduces a dynamic approach in which the urgency of detection is modulated based on the system's proximity to unsafe operational states. The core detection mechanism relies on a CUSUM-based algorithm that evaluates residuals—differences between observed and predicted sensor values—derived from a hybrid CNN-RNN behavior predictor. By adjusting the drift parameter, the system can dynamically prioritize either faster detection or reduced false positives, depending on the operational context. This work is directly relevant to measuring the TTD, as it explicitly models and controls the detection delay through a drift adaptor that estimates a detection deadline. It also distinguishes between stateless strategies (which raise alerts on immediate deviation and often inflate false positives) and stateful strategies (which rely on sustained deviations to trigger alerts), both of which significantly affect the time between the actual onset of an attack and the issuance of an alert. This nuanced approach supports the idea that measuring TTD should incorporate not only algorithmic latency but also system-dependent thresholds that govern alert responsiveness. However, it has not considered the delay between the attack started and the time that IDS collected the attack logs.

Lin et al. [59] provided a comprehensive overview of six publicly available ICS datasets—such as SWaT and BATADAL—and demonstrate their use in both research and education. Their TABOR framework leverages Timed Automata and Bayesian Networks to model and detect anomalies in physical process data, achieving explainable, high-accuracy detection on the SWaT testbed. While this work highlights the value of realistic ICS datasets for developing and teaching detection

techniques, it does not evaluate the temporal performance of these methods. In particular, it omits key timing considerations—such as the delay between attack onset and log availability, and the latency incurred in raising an alert after detection—which are essential for assessing an IDS's true end-to-end responsiveness in operational environments.

Tsiami and Makropoulos [60] proposed a one-stage anomaly detection method based on Temporal Graph Convolutional Networks (TGCNs) for detecting cyber–physical attacks in water distribution systems (WDS). Their approach models the WDS as a graph, capturing both spatial and temporal correlations from SCADA sensor data. Anomalies are identified using the Mahalanobis distance between predicted and observed sensor values. To evaluate detection performance, they introduced a metric called $S_{TTD}$ , which measures the detection delay as a fraction of the total attack duration. While this metric provides a useful measure of relative detection speed, the study does not assess the absolute time elapsed from the onset of the attack to the generation of an alert. In particular, the evaluation omits delays such as the time required to collect attack data or to raise an alert after detection—elements that are critical for capturing the full end-to-end responsiveness of intrusion detection systems.

To better illustrate the extent to which existing studies consider the different components of detection time, a comparative summary is provided in Table 4.1. This table highlights the aspects of detection timing that each work evaluated and which critical delays were overlooked in relation to a comprehensive time-to-detection perspective.

## 4.3 Model of TTD

The detection process comprises several interconnected steps, starting with data acquisition and continuing through local detection, communication time, and global processing. Here, we explain each step completely.

### 4.3.1 Data Aquisition

This stage refers to the initial phase of TTD, where raw data is collected from the physical layer in the grid by using sensing devices. Voltage transformers and current transformers record physical

Table 4.1: Comparison of Related Works Based on TTD Coverage

| Study | What Was Measured | Missed Aspects Related to TTD |
|---|---|---|
| Zhou et al. [50] | Model inference speed and classifier runtime after feature embedding (using DFEL on NSL-KDD and UNSW-NB15 datasets) | Did not account for the time between attack onset and IDS log collection, or the delay between detection and alert generation. |
| Ozalp and Albayrak [57] | Classifier runtime (RF, J48, NB, MLP) after applying feature selection on NSL-KDD; focused on reducing processing delay | Did not include the delay before the IDS receives the attack logs or the time required to raise an alert after detection. |
| Akowuah et al. [58] | Modeled and adapted detection delay through a dynamic drift controller based on system state urgency | Did not consider the delay between the actual attack initiation and when logs are first received by the IDS. |
| Lin et al. [59] | Proposed an anomaly detection framework (TABOR) using public ICS datasets; focused on explainability and dataset applicability | Did not measure detection timing; omitted log collection delay and post-detection alerting latency. |
| Tsiami and Makropoulos [60] | Proposed $S_{\text{TTD}}$ as a metric to evaluate detection delay relative to attack duration using a TGCN-based method in WDS | Did not measure absolute delay from attack onset to alert; omitted log collection time and alert generation latency. |

measurements in real-time, such as current and voltage, thus acting as the primary reference basis for attack detection. However, the raw measurements do not directly communicate with the IDS; instead, they are conveyed via intermediate devices and systems, which causes delays and affects the overall TTD. Specifically, devices such as Intelligent Electronic Devices (IEDs) and Remote Terminal Units (RTUs) preprocess and digitize sensor data, thereby adding to the delay in the log collection process. Thereafter, interfaces such as SCADA become important for acquiring and transmitting data back to the control center or IDS, which can cause additional transmission delays. By defining the roles played by sensors, equipment, and systems in this manner, the model explains the contribution of each layer with respect to the duration between the start of the attack and its detection, providing grounds for measuring and minimizing TTD.

### 4.3.2 Communication Time

The communication time refers to the time it takes to transfer sensor data or local detection results to a central monitoring or analysis system. This process commences when the data is available

in its raw form and continues through to when it reaches the IDS engine for analysis. How communication takes place here also plays a crucial role in this phase, operating efficiently. For instance, IEC 61850[61], designed for rapid data exchange between electrical substations, and DNP3[62], which supplies secure communication in the majority of SCADA systems for remote monitoring.

### 4.3.3 Local Detection

In the proposed TTD model, local detection plays a pivotal role in reducing the detection delay by performing preliminary analysis near the data source. After raw measurements are collected from sensors and processed by intermediary devices, local detection represents the first computational step toward identifying an ongoing cyberattack, corresponding to the calculation phase in the IDS timeline. At this phase, ML algorithms process consolidated telemetry data to validate anomalies. This detection typically occurs within IEDs or Phasor Measurement Units (PMUs), which are embedded with lightweight or rule-based algorithms capable of flagging anomalies without requiring data to be transmitted to a central system.

### 4.3.4 Global Processing

This stage involves centralized systems that aggregate and analyze data from distributed sources, particularly focusing on the final step in the TTD model: **alert generation**. In the defined TTD framework, this step corresponds to the detection system issuing a formal alert after completing the computation. The latency between computation time and raising for alert time reflects the alert transmission time, a crucial factor in industrial networks that is often overlooked in many academic evaluations.

The centralized monitoring infrastructure, commonly implemented as a Security Operations Center (SOC), plays a pivotal role by correlating logs, prioritizing alerts, and initiating responses. This function aligns with the system architecture and guidance presented in *NIST SP 1800-7* [63], which emphasizes real-time situational awareness and anomaly response in electric utilities. Furthermore, the *MITRE white paper on best practices for SOCs* [64] highlights that a well-functioning SOC should aim to minimize delays from detection to alerting, thereby supporting swift mitigation.

Thus, this stage encapsulates both the final delay in the TTD model and the operational effectiveness of the alerting mechanism.

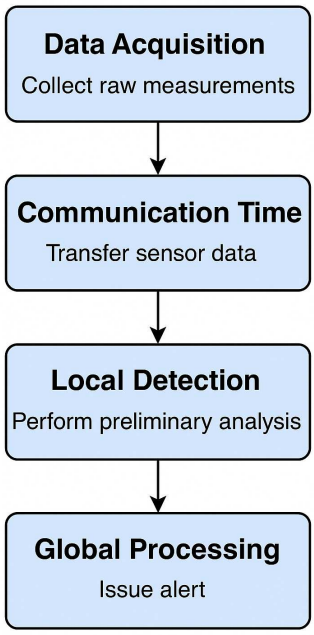


Figure 4.1: Workflow of Time to Detection

## 4.4 Problem Formulation

### 4.4.1 What is the TTD?

Most existing research defines TTD as the period from the moment an IDS receives attack data to the point at which it detects the attack. This definition overlooks earlier stages in the attack timeline, such as when the attack actually begins and the delays introduced by communication and system processing. As a result, current approaches often underestimate the total time it takes to respond to a threat in real-world environments. There is a clear need to redefine TTD in a manner that encompasses all relevant delays and more accurately reflects the practical detection timelines in industrial systems. In practice, network latencies and processing delays at control centers significantly impact detection times. Many studies provide an incomplete understanding of system performance. By failing to account for the holistic workflow, these studies risk underestimating the

practical challenges involved in timely cyberattack detection.

### 4.4.2 How to properly measure the TTD?

In addition to the lack of clarity around the role of ML methods in detection pipelines, another fundamental gap in the literature is the absence of a consistent or practical approach to measuring the TTD. While many existing works report detection performance metrics such as accuracy or precision, they either oversimplify or entirely overlook the temporal aspect of when an attack is detected relative to when it begins. Without a clear and standardized method for capturing this timing, it becomes challenging to assess the real-world applicability of detection systems, particularly in time-sensitive environments such as ICSs.

## 4.5 Methodology

To evaluate and analyze the responsiveness of IDSs in ICS, this thesis introduces a structured TTD model that decomposes the detection timeline into discrete and explainable components. As illustrated in Figure 4.2, the model is defined along two parallel temporal axes: the Attack Axis and the IDS Axis. The Attack Axis traces the actual sequence of events initiated by an attacker, while the IDS Axis reflects the internal sequence of detection-related operations carried out by the IDS infrastructure. This dual-axis representation enables a clear distinction between the cause (the attack event) and the effect (the detection and alert process).

The model defines a more accurate measurement of TTD as the time elapsed from the initiation of the cyberattack to the moment when an alert is ready to be issued by the IDS. Mathematically, it is captured by the duration from $T_0$ (attack initiation time) to $T_4'$ (alert issuance time). The full timeline explains in the next part.

### 4.5.1 Attack Axis vs. IDS Axis

The Attack Axis represents the real-world initiation of a cyberattack. In operational systems, this could correspond to the deployment of a malicious payload or the execution of any unauthorized command. However, this attack does not immediately become observable to the IDS. The

system's physical or digital sensors must first perceive abnormal behaviors, which are then relayed to monitoring systems.

The IDS Axis begins with the moment when the system starts receiving log data that may contain evidence of the attack ($T_0'$). This log collection is not instantaneous. Many ICS components, especially those under SCADA systems, operate on periodic sampling intervals (e.g., every 100–500 ms), and logs are often batched or queued. This introduces an initial delay between the attack's occurrence and the IDS's first chance to observe it.

### 4.5.2 Components of TTD

- $T_0$**: Attack Initiation**

  The attack initiation time marks the beginning of the adversarial activity. In real-world scenarios, determining this point is challenging due to the stealthy nature of cyberattacks. However, in benchmark datasets like SWaT or BATADAL, this timestamp is explicitly annotated, allowing it to serve as a reliable reference.

- $T_0'$**: Log Collection**

  After the attack begins, there is an inevitable delay before any related data is collected. This delay arises due to communication latencies and waiting times within the SCADA component, like sensors. Even if an attack has started, the system may only log its effects during the next cycle or time slot. Consequently, $T_0'$ is the first point in time where the IDS has any chance of detecting the attack, making it a lower bound for observable evidence.

- $T_1$**: Attack End Time**

  This is the point at which the attack physically or logically ends from the attacker's perspective. For instance, the attacker stops injecting malicious commands or interfering with system operations. This is also marked in benchmark datasets, which facilitates defining the full attack window.

- $T_1'$**: Attack Log End Time in IDS**

  This is the last log entry that contains data impacted by the attack. It corresponds to the final

observable effects of the attack. This point may lag behind the true attack end time $T_1$ due to system inertia or delayed propagation effects.

- $T_2$**: Appropriate Required Time For Response**

  To assess the timeliness of detection, this thesis introduces a detection deadline variable $T_2$, which defines the maximum acceptable delay between the start of an attack and the generation of an alert. The specific value of $T_2$ depends on the timing requirements of the application domain, such as SCADA-based systems or PMU-based systems. It will be selected accordingly during the experiment phase.

- $T_2'$**: Transmission to Analysis Component**

  After log preprocessing, the data must be transmitted to the IDS's analysis component—typically located at a central monitoring server. Depending on the network architecture (e.g., if using remote servers, cloud-based analytics, or segmented LANs), this transmission can incur non-trivial delays, especially under bandwidth constraints or heavy traffic conditions.

- $T_3'$**: Detection Completion**

  At this stage, the IDS has completed its anomaly or signature-based analysis. If machine learning or deep learning models are used (such as CNN-LSTM), this time includes the full duration of model inference. The detection engine makes a binary or probabilistic decision regarding whether an attack is present in the input data.

- $T_4'$**: Alert Generation**

  Once a detection decision is made, the system may still need time to format and transmit the alert to appropriate response systems or dashboards. This step may involve setting confirmation thresholds, implementing rule-based escalation, or utilizing alert throttling mechanisms to prevent spamming. As a result, there is often a short but meaningful gap between $T_3'$ and $T_4'$.

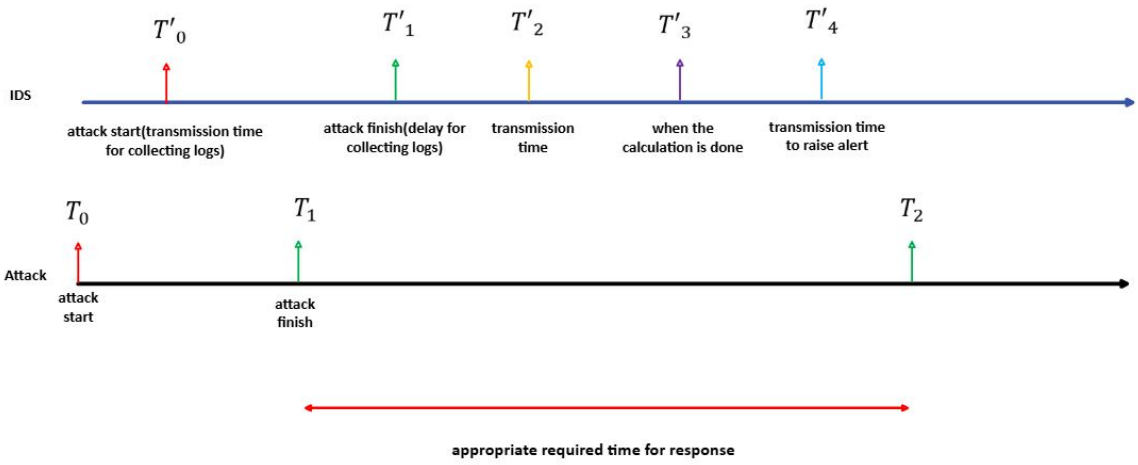


Figure 4.2: Model of Time to Detection Process

## 4.6 Applying The TTD

To evaluate and demonstrate the relevance of the proposed TTD model, we designed an experiment using the SWaT dataset, which provides time-stamped sensor readings from a water treatment ICS testbed. The timestamp column in SWaT is used to establish the actual start time of an attack ($T_0$). This allows us to align the true onset of malicious activity with the detection pipeline and evaluate the full detection timeline.

### 4.6.1 Machine Learning Model Description

The thesis implemented a hybrid CNN-LSTM model for binary classification to detect normal and attack instances in the SWaT dataset. The selection of this architecture is motivated by the unique characteristics of ICS environments, where cyberattacks often manifest through subtle spatial patterns and evolve with delayed impacts on system behaviour. CNNs [65] are employed to extract local spatial features from multivariate sensor data, identifying correlations across different system variables at each time step. LSTM networks [65] are then used to model long-range temporal dependencies, capturing the progression and buildup of abnormal patterns over time, as they have their own memories and can make relatively accurate forecasting. This hybrid approach is particularly well-suited for intrusion detection in ICS, where both instantaneous anomalies and their temporal evolution are critical for early and accurate attack detection. By leveraging the strengths

of both CNN and LSTM components, the model provides a more robust framework for detecting sophisticated, time-dependent attack scenarios in real-world industrial settings.

The model architecture includes a 1-dimensional convolutional layer with 64 filters, a kernel size of 1, and ReLU activation, followed by an LSTM layer with 64 units, a dense layer with 64 units and ReLU activation, and a final output layer with softmax activation. We used the Adam optimizer and categorical cross-entropy loss. Before training, all features were normalized using MinMaxScaler to scale them to the [0, 1] range, and the input was reshaped into a 3D format with a time step of 10 to suit the CNN-LSTM structure. The model was trained with a 20% validation split from the training set.

### 4.6.2 Detection Time Mapping

In the framework of assessing the timeliness of intrusion detection systems for ICS, the response time depends on several factors, such as the type of incident, the criticality of the resources and data that are affected, the severity of the incident, the time and day of the week, and other incidents that the team is handling. Generally, the highest priority is handling incidents that are likely to cause the most damage to the organization or to other organizations [66]. According to IEEE Standard C37.118.1 [67], the detection deadline for PMU-based datasets is much shorter than SCADA-based datasets, because PMUs operate at high sampling rates and are used in real-time monitoring of power grid dynamics. The appropriate detection deadline for PMU-based datasets would be between 20 ms and 200 ms. For the purposes of this experiment, we assign a detection deadline of $T_2 = 1000$ ms, based on typical latency tolerances observed in SCADA-based water treatment systems. As the dataset is SCADA-based data, this value represents the maximum allowed time from the beginning of the initiation phase of a cyberattack ($T_0$) to the moment when an alert is raised ($T_4'$). Although current industry standards do not formally require detection deadlines, guidelines such as NIST SP 800-82 Rev. 2 [68] emphasize the importance of timely detection and response actions to minimize, as much as possible, potential negative impacts on ICS operations. The choice of a 1000 ms deadline is consistent with these guidelines. It represents a reasonable benchmark that balances the pressing need for fast detection with the inevitable communication and processing delays inherent in SCADA systems.

To evaluate the IDS performance, we measure the TTD by calculating the time it takes from the beginning of an attack until the system raises an alert.

To empirically evaluate specific components of the proposed TTD model, a simplified SCADA testbed was implemented using the OMNeT++ discrete-event network simulator. The objective was to simulate the transmission of sensor data through an IDS component before reaching a PLC and to measure the latency for raising an alert in case an attack is detected, thereby allowing the measurement of communication latency between these key nodes in a control system. The simulation has been designed to add the transmission time to the timestamps of the data's beginning time.

The simulation network includes four main modules:

- **SensorNode**: Periodically generates messages (800 bits) to simulate sensor output. Each message is timestamped at the point of generation to capture its initial transmission time.
- **IDS Module**: Receives incoming packets from the sensor, simulates a lightweight intrusion detection process, and determines whether the message indicates a potential attack. If an attack is detected, the IDS forwards a corresponding alert to the AlertSink; otherwise, the packet is relayed to the PLC as a normal message.
- **PLC**: Acts as the standard destination that logs and acknowledges the receipt of normal messages.
- **AlertSink**: Serves as the terminal for alerts raised by the IDS. This module logs the time of alert issuance, which is used to calculate the interval between detection and response, corresponding to $T_4'$.

The modules are interconnected using a custom `RealisticChannel`, which extends OMNeT++'s `DatarateChannel`. It is configured with a transmission rate of 100 kbps. This setup realistically simulates the communication constraints present in industrial SCADA systems, where delays can impact both sensor-to-controller transmission and the propagation of alerts.

Figure 4.3 illustrates the simulated SCADA network topology and the corresponding OMNeT++ timeline of message and alert propagation.

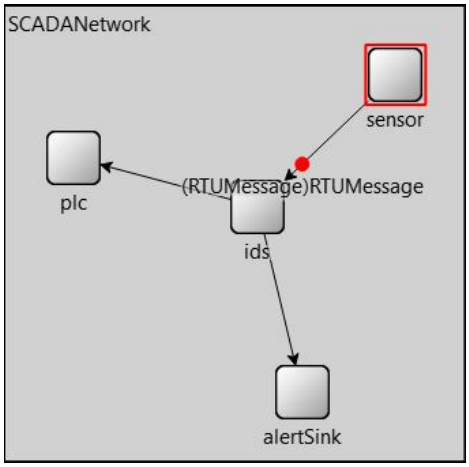


Figure 4.3: Simulation of SCADA Network By OMNET++

During the simulation, the delay of each message was recorded from its generation at the SensorNode to its arrival at both the IDS and the AlertSink modules. This setup allowed for the isolation and measurement of two critical timing components: the transmission delay from the sensor to the IDS ($T_2' - T_1'$) and the time required to raise an alert after detection ($T_4' - T_3'$). By separating these intervals, the simulation provides a more accurate representation of SCADA communication behavior under real-world timing constraints. The observed delays were then incorporated into the analytical experiment on the SWaT dataset, enabling a more realistic evaluation of the full TTD as defined by the proposed model. Thus, the TTD is calculated as TTD $= T_4' - T_0$, representing the time elapsed between attack initiation and alert generation. This duration includes multiple delay components such as log collection delay, transmission delay, IDS detection time, and alert generation latency, as summarized in Table 4.2.

### 4.6.3 Use Case and Significance of the Model

Prior researches often report only algorithmic runtime or prediction delay ($T_3'$ - $T_2'$). Existing literatures often just mention the TTD from the time that data reaches the IDS to be analyzed by the detection system ($T_2'$) until the system detects the cyberattack ($T_3'$); however, this model decomposes the detection process into discrete stages that mirror real-world IDS operations. Each stage introduces potential latency, and overlooking any component leads to an incomplete understanding of IDS responsiveness. For example, measuring only inference time ignores the overhead introduced

Table 4.2: Mapping of Detection Time Components in the Proposed TTD Model

| Variable | Description | How It Is Handled |
|---|---|---|
| $T_0$ | Actual time the attack starts | Taken directly from the `timestamp` column in the SWaT dataset |
| $T_0'$ | Time the attack enters the IDS | Time when the attack logs received by sensors |
| $T_2'$ | Transmission time | Time to send message from sensors/actuators to the IDS |
| $T_3'$ | Time the IDS finishes detection | Time when the model completes processing and detects the attack |
| $T_4'$ | Time the alert is raised | Simulated as $T_3'$ plus a random delay assumed SCADA delay |
| TTD | Proposed Detection Time | Computed as $T_4' - T_0$ |

by polling intervals, communication bottlenecks, or decision propagation. By explicitly modeling each of these, the proposed TTD framework provides a practical and operationally grounded method for evaluating and improving detection systems in ICS environments.

This TTD model serves as the methodological foundation for evaluating detection performance in existing intrusion detection approaches. By aligning the detection delay ($T_4'$ - $T_0$) with the critical response deadline ($T_2$ - $T_0$), the model enables the evaluation of whether an IDS can respond in time to prevent physical or operational damage in SCADA environments ($T_2$ - $T_1$). If the time the alert raises ($T_4'$) is after the appropriate response time ($T_2$), it can lead to harmful damage.

## 4.7 Results

In this section, we outline the experimental procedure undertaken to evaluate the TTD of cyber-attacks in ICS using the proposed TTD model. The experiment was designed to apply this model in an offline environment using the SWaT dataset, allowing for precise control over data handling and model evaluation. As the implementation is not conducted in a real-time SCADA system, there is no delay between the actual occurrence of the attack and the time the log data is collected. Therefore, we assume that the attack start time ($T_0$) and the timestamp at which the IDS receives the log ($T_0'$) are identical in this setup, i.e., $T_0 = T_0'$.

### 4.7.1 Evaluation Process

Each attack instance in the SWaT test set was evaluated independently using the implemented CNN-LSTM model. The dataset was split into training and testing sets using an 80:20 ratio, and the model's predictions over the test set were continuously monitored. For each attack sample, the earliest timestamp at which the IDS successfully flagged malicious behavior was recorded as $T_3'$ (detection completion time).

To incorporate the transmission delay that occurs between the time when log data becomes available ($T_0'$) and when it reaches the IDS analysis component ($T_2'$), we used the SCADA simulation described in the previous section. The simulation modeled a typical SCADA message path from sensor to IDS, depending on whether it received normal or attack data. It then proceeded to the PLC device or raised an alert for attack detection. The transmission delay and time for raising an alert measured across samples were added to the original log timestamp $T_0'$ of each attack instance in the SWaT dataset. This adjusted timestamp represents a more accurate approximation of when the IDS receives the data for analysis.

We adopted an exemplary detection deadline of 1,000 ms based on the timing tolerances provided in NIST SP 800-82 Rev. 2 guidelines for ICS [68]. This approach enables a more realistic and comprehensive evaluation of detection timeliness, accounting for both network-induced and system-induced delays.

Table 4.3: Proposed TTD Measurements for Selected SWaT Attacks

| **Attack #** | **Start Time** ($T_0$) | **Transmission Time** ($T_2'$) | **Detected Time** ($T_3'$) | **Alert Time** ($T_4'$) | **TTD** ($T_4' - T_0$) |
|---|---|---|---|---|---|
| 1 | 14:59:17.161 | 14:59:17.172 | 14:59:17.370 | 14:59:17.763 | 602 ms |
| 2 | 14:59:17.359 | 14:59:17.369 | 14:59:17.435 | 14:59:17.578 | 218 ms |
| 3 | 14:59:38.762 | 14:59:38.773 | 14:59:38.843 | 14:59:39.235 | 473 ms |
| 4 | 14:59:40.017 | 14:59:40.030 | 14:59:40.151 | 14:59:40.432 | 415 ms |
| 5 | 15:01:12.258 | 15:01:12.269 | 15:01:12.390 | 15:01:12.776 | 518 ms |
| 6 | 15:01:13.727 | 15:01:13.737 | 15:01:13.823 | 15:01:14.123 | 396 ms |
| 7 | 15:01:14.880 | 15:01:14.890 | 15:01:14.957 | 15:01:15.266 | 386 ms |
| 8 | 15:23:39.061 | 15:23:39.073 | 15:23:39.701 | 15:23:39.886 | 825 ms |
| 9 | 15:23:54.008 | 15:23:54.021 | 15:23:54.669 | 15:23:54.982 | 974 ms |
| 10 | 15:25:11.271 | 15:25:11.285 | 15:25:11.407 | 15:25:11.578 | 306 ms |

Table 4.3 presents the measured TTD values for ten representative cyberattack instances, based on the refined TTD model. Each row breaks down the detection timeline into four key timestamps: attack start time ($T_0$), transmission time to the IDS ($T_2'$), detection completion time by the CNN-LSTM model ($T_3'$), and the time the alert is raised ($T_4'$). The delay from $T_0$ to $T_2'$ represents the communication latency from the sensor to the IDS (i.e., transmission delay), which ranges from 10 ms to 15 ms across the attacks. The interval between $T_2'$ and $T_3'$ captures the internal inference time of the CNN-LSTM detection model. The gap from $T_3'$ to $T_4'$ represents the alert propagation delay to the AlertSink, which varies between 100 ms and 500 ms in this simulation. The total TTD, calculated as $T_4' - T_0$, varies from 218 ms to 974 ms, indicating that the model captures both fast and relatively slower detection scenarios.

Unlike prior research that typically considers only the time from attack start to detection (or ignores exact timing altogether), this study decomposes the detection delay into four distinct and measurable components: the attack start time, the transmission time, the detection time, and the alert time. By simulating a simple SCADA system to measure transmission delay and latency, and raise an alert in case an attack is detected, this model provides a more comprehensive representation of the operational timeline of an IDS in real-world scenarios. Previous works often overlook such post-detection overheads, which can be non-negligible in practice, and thus may underestimate the

system's actual responsiveness.

Importantly, all attack instances presented in this evaluation were successfully detected and alerted within a detection deadline of 1,000 ms. This supports the feasibility of using the proposed TTD framework as a robust and measurable performance metric in developing and benchmarking ML-based IDS for ICS. By accounting for practical timing elements—such as system latency and alert processing time—the model advocates for more realistic and standardized TTD reporting in future research. We argue that the more accurate TTD should be formally considered alongside conventional metrics, such as accuracy and F1-score, as it directly reflects an IDS's ability to meet the real-time constraints critical to protecting cyber-physical infrastructure.

### 4.7.2 Conclusion

This chapter presents the new model of TTD and its implementation using the proposed TTD model with the SWaT dataset under a CNN-LSTM architecture. By identifying the start times of the attacks, transmission delays, detection moments, and measuring the delay for alert generation, the proposed TTD was measured for multiple attack instances. The experimental setup was designed to simulate offline analysis conditions, where the start time of the attack ($T_0$) coincides with the initial timestamp of log collection ($T_0'$). The transmission delay has been added to the experiment by simulating a simple SCADA system to measure the transmission time. The hybrid CNN-LSTM model was employed to learn the temporal-spatial characteristics of the data. Then, the time to detection of attacks, as determined by the CNN-LSTM model, was measured. Finally, the delay in raising an alert was added by a simulated model implemented in OMNET++. The proposed TTD was performed well, and the recorded times were all within the 1000 ms deadline in all cases.

The TTD results provide empirical insights into the responsiveness of the detection system in ICS environments. This framework provides a foundation for future work that could incorporate online environments, dynamic alert delay estimation, or more granular categorization of attack types. The chapter also reinforces the need for detection-time-aware evaluations in cyber-physical system security, emphasizing that timeliness is just as critical as accuracy in assessing IDS effectiveness.

# Chapter 5

# Conclusion

The rising frequency and sophistication of cyberattacks on critical infrastructures have intensified the need for robust IDS. Particularly in ICS and smart grid environments, where operational continuity is vital, the ability to detect attacks promptly is essential. While IDS technologies have matured, their effectiveness still depends heavily on the quality and balance of the data they are trained and evaluated on, as well as the metrics used to measure their performance. Among these, TTD has emerged as a crucial factor, especially in OT systems where the priority is often availability over confidentiality. Hence, this thesis focuses on two central challenges in evaluating ML-based IDSs: (1) class imbalance in cybersecurity datasets, and (2) accurate modeling and measurement of TTD.

To explore these challenges, the thesis first categorized and reviewed existing cybersecurity datasets by separating them into two domains—IT and OT—based on their operational environments. Several public IT and OT datasets were discussed. A key observation from this review was that many existing datasets suffer from class imbalance and often lack critical elements such as timestamp precision, making the realistic evaluation of IDSs difficult. Based on these insights, the thesis presents two case studies: one that empirically examines the impact of resampling techniques for handling class imbalance across IT and OT datasets, and another that introduces and applies a comprehensive TTD model to capture more realistic detection delays in ICS environments.

This thesis examined the effectiveness of various resampling techniques in addressing class imbalance within cybersecurity datasets, focusing on two scenarios: when only the training data

is balanced, and when both the training and testing data are balanced. Rather than proposing a novel resampling method, the objective was to empirically assess the impact of existing undersampling, oversampling, and hybrid sampling strategies on the performance of deep learning models applied to IT (CICIDS2017) and OT (SWaT) datasets. The findings show that oversampling, particularly SMOTE, generally yields better or more stable performance—especially in small, time-series datasets like SWaT—when only training data is balanced. However, balancing both training and testing sets further improves results in large-scale datasets, such as CICIDS2017. These insights emphasize that the choice of resampling method should be informed by the dataset size, structure, and application context, and that balancing strategies must be aligned with real-world deployment goals to avoid inflated or misleading performance metrics.

This thesis also investigated the accurate measurement of TTD for cyberattacks in ICS, addressing critical gaps in existing literature. While prior research often defines TTD only from the time a system receives an attack log until the IDS completes processing or misses some other parts of measuring true TTD, this simplified view omits important delay components. Specifically, delays such as log collection time and alert-raising latency are typically neglected, leading to an incomplete assessment of detection capabilities. To address this, we introduced a more comprehensive model of true TTD, incorporating all key stages from the actual start of an attack to the point at which an alert is raised. To evaluate this model, we implemented a CNN-LSTM-based IDS on the SWaT dataset. The timestamp column was used to mark the true attack initiation time, while the model's detection output defined the processing completion point. To simulate realistic operational conditions, a random artificial delay was introduced to approximate the time required to raise an alert, emulating SCADA response behavior. The result is a more holistic and accurate representation of TTD, which better reflects real-world performance and sets a foundation for benchmarking future detection systems in ICS environments.

Looking ahead, future research should focus on evaluating resampling techniques better. We will investigate emerging Generative AI approaches (e.g., GAN- or VAE-based augmentation) as alternative solutions for class imbalance, comparing their impact against traditional resampling methods. Also, there would be a need to find appropriate ways to reduce the TTD, especially the runtime of ML models. Moreover, further validation of the TTD model across other OT datasets

would reinforce its applicability. Expanding this work to consider false alarm reduction, early detection under concept drift, and the integration of real-time response strategies would significantly advance the robustness and utility of IDS in critical infrastructure protection.

# Bibliography


[1] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Félix J García Clemente, Cristian Cadenas Sarmiento, Carlos Javier Del Canto Masa, and Rubén Méndez Nistal. On the generation of anomaly detection datasets in industrial control systems. *IEEE Access*, 7:177460–177473, 2019.

[2] Aziz Naanani and Noureddine Masaif. Security in industry 4.0: Cyber-attacks and countermeasures. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(10): 6504–6512, 2021.

[3] Preeti Mishra, Vijay Varadharajan, Uday Tupakula, and Emmanuel S Pilli. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE communications surveys & tutorials*, 21(1):686–728, 2018.

[4] Marek Pawlicki, Michał Choraś, Rafał Kozik, and Witold Hołubowicz. On the impact of network data balancing in cybersecurity applications. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part IV 20*, pages 196–210. Springer, 2020.

[5] Wm Arthur Conklin. IT vs. OT security: A time to consider a change in cia to include resiliency. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 2642–2647. IEEE, 2016.

[6] Glenn Murray, Michael N Johnstone, and Craig Valli. The convergence of it and ot in critical infrastructure. 2017.

[7] Nishchal Singh Kush, Ejaz Ahmed, Mark Branagan, and Ernest Foo. Poisoned goose: Exploiting the goose protocol. In *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)[Conferences in Research and Practice in Information Technology, Volume 149]*, pages 17–22. Australian Computer Society, 2014.

[8] Juan Hoyos, Mark Dehus, and Timthy X Brown. Exploiting the goose protocol: A practical attack on cyber-infrastructure. In *2012 IEEE Globecom Workshops*, pages 1508–1513. IEEE, 2012.

[9] Sinil Mubarak, Mohamed Hadi Habaebi, Md Rafiqul Islam, Farah Diyana Abdul Rahman, and Mohammad Tahir. Anomaly detection in ics datasets with machine learning algorithms. *Computer Systems Science & Engineering*, 37(1), 2021.

[10] Mohamad Kaouk, Jean-Marie Flaus, Marie-Laure Potet, and Roland Groz. A review of intrusion detection systems for industrial control systems. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1699–1704. IEEE, 2019.

[11] Johan Angséus and Rikard Ekbom. Network-based intrusion detection systems for industrial control systems. 2017.

[12] Seungoh Choi, Jeong-Han Yun, and Sin-Kyu Kim. A comparison of ics datasets for security research based on attack paths. In *Critical Information Infrastructures Security: 13th International Conference, CRITIS 2018, Kaunas, Lithuania, September 24-26, 2018, Revised Selected Papers 13*, pages 154–166. Springer, 2019.

[13] William Stallings. Cryptography and network security: principles and practices, 2005.

[14] Manu Bijone. A survey on secure network: intrusion detection & prevention approaches. *American Journal of Information Systems*, 4(3):69–88, 2016.

[15] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Christos Tachtatzis, and Robert Atkinson. Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv preprint arXiv:1701.02145*, 2017.

[16] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). Technical report, NIST Special Publication 800-94, 2007.

[17] Mohiuddin Ahmed, Azad Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2016.

[18] Seba Anna Varghese, Alireza Dehlaghi Ghadim, Ali Balador, Zahra Alimadadi, and Panos Papadimitratos. Digital twin-based intrusion detection for industrial control systems. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 611–617. IEEE, 2022.

[19] Muhammad Azmi Umer, Khurum Nazir Junejo, Muhammad Taha Jilani, and Aditya P Mathur. Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations. *International Journal of Critical Infrastructure Protection*, 38: 100516, 2022.

[20] Sydney Mambwe Kasongo and Yanxia Sun. A deep learning method with filter based feature engineering for wireless intrusion detection system. *IEEE access*, 7:38597–38607, 2019.

[21] Xu Yang and Zhao Hui. Intrusion detection alarm filtering technology based on ant colony clustering algorithm. In *2015 Sixth International Conference on Intelligent Systems Design and Engineering Applications (ISDEA)*, pages 470–473. IEEE, 2015.

[22] Urslla Uchechi Izuazu, Vivian Ukamaka Ihekoronye, Dong-Seong Kim, and Jae Min Lee. Securing critical infrastructure: A denoising data-driven approach for intrusion detection in ics network. In *2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 841–846. IEEE, 2024.

[23] Oyeniyi Akeem Alimi, Khmaies Ouahada, Adnan M Abu-Mahfouz, Suvendi Rimer, and Kuburat Oyeranti Adefemi Alimi. A review of research works on supervised learning algorithms for scada intrusion detection and classification. *Sustainability*, 13(17):9597, 2021.

[24] Abigail MY Koay, Ryan K L Ko, Hinne Hettema, and Kenneth Radke. Machine learning

in industrial control system (ics) security: current landscape, opportunities and challenges. *Journal of Intelligent Information Systems*, 60(2):377–405, 2023.

[25] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[26] Robert Koch. Towards next-generation intrusion detection. In *2011 3rd International Conference on Cyber Conflict*, pages 1–18. IEEE, 2011.

[27] Ankit Thakkar and Ritika Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167:636–645, 2020.

[28] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[29] Hossein Hadian Jazi, Hugo Gonzalez, Natalia Stakhanova, and Ali A Ghorbani. Detecting http-based application layer dos attacks on web servers in the presence of sampling. *Computer Networks*, 121:25–36, 2017.

[30] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[31] Richard P Lippmann, David J Fried, Isaac Graf, Joshua W Haines, Kristopher R Kendall, David McClung, Dan Weber, Seth E Webster, Dan Wyschogrod, Robert K Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 12–26. IEEE, 2000.

[32] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[33] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167, 2019.

[34] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In *Annual Computer Security Applications Conference*, pages 480–495, 2020.

[35] Guowei Shen, Wanling Wang, Qilin Mu, Yanhong Pu, Ya Qin, and Miao Yu. Data-driven cybersecurity knowledge graph construction for industrial control system security. *Wireless Communications and Mobile Computing*, 2020:1–13, 2020.

[36] Sayawu Yakubu Diaba, Theophilus Anafo, Lord Anertei Tetteh, Michael Alewo Oyibo, Andrew Adewale Alola, Miadreza Shafie-Khah, and Mohammed Elmusrati. Scada securing system using deep learning to prevent cyber infiltration. *Neural Networks*, 2023.

[37] Aida Ali, Siti Mariyam Shamsuddin, and Anca L Ralescu. Classification with class imbalance problem. *Int. J. Advance Soft Compu. Appl*, 5(3):176–204, 2013.

[38] Bakht Sher Ali, Inam Ullah, Tamara Al Shloul, Izhar Ahmed Khan, Ijaz Khan, Yazeed Yasin Ghadi, Akmalbek Abdusalomov, Rashid Nasimov, Khmaies Ouahada, and Habib Hamam. Ics-ids: application of big data analysis in ai-based intrusion detection systems to identify cyberattacks in ics networks. *The Journal of Supercomputing*, 80(6):7876–7905, 2024.

[39] Gozde Karatas, Onder Demir, and Ozgur Koray Sahingoz. Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. *IEEE access*, 8:32150–32162, 2020.

[40] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36, 2004.

[41] Asaad Balla, Mohamed Hadi Habaebi, Elfatih AA Elsheikh, Md Rafiqul Islam, and FM Suliman. The effect of dataset imbalance on the performance of scada intrusion detection systems. *Sensors*, 23(2):758, 2023.

[42] Sikha Bagui and Kunqi Li. Resampling imbalanced data for network intrusion detection datasets. *Journal of Big Data*, 8(1):6, 2021.

[43] Ngan Tran, Haihua Chen, Janet Jiang, Jay Bhuyan, and Junhua Ding. Effect of class imbalance on the performance of machine learning-based network intrusion detection. *International Journal of Performability Engineering*, 17(9):741, 2021.

[44] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[45] Gustavo EAPA Batista, Andre CPLF Carvalho, and Maria Carolina Monard. Applying one-sided selection to unbalanced datasets. In *Mexican International Conference on Artificial Intelligence*, pages 315–325. Springer, 2000.

[46] Lian Yu and Nengfeng Zhou. Survey of imbalanced data methodologies. *arXiv preprint arXiv:2104.02240*, 2021.

[47] Jingmei Liu, Yuanbo Gao, and Fengjie Hu. A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm. *Computers & Security*, 106:102289, 2021.

[48] Abebe Tesfahun and D Lalitha Bhaskari. Intrusion detection using random forests classifier with smote and feature reduction. In *2013 International conference on cloud & ubiquitous computing & emerging technologies*, pages 127–132. IEEE, 2013.

[49] Kaiyuan Jiang, Wenya Wang, Aili Wang, and Haibin Wu. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE access*, 8:32464–32476, 2020.

[50] Hongpo Zhang, Lulu Huang, Chase Q Wu, and Zhanbo Li. An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks*, 177:107315, 2020.

[51] Zhe Deng, Ants Torim, Sadok Ben Yahia, and Hayretdin Bahsi. Generative ai in intrusion detection systems for internet of things: A systematic literature review. *IEEE Open Journal of the Communications Society*, 2025.

[52] Xu Guo and Yiqiang Chen. Generative ai for synthetic data generation: Methods, challenges and the future. *arXiv preprint arXiv:2403.04190*, 2024.

[53] Saifur Rahman, Shantanu Pal, Shubh Mittal, Tisha Chawla, and Chandan Karmakar. Syn-gan: A robust intrusion detection system using gan-based synthetic data for iot security. *Internet of Things*, 26:101212, 2024.

[54] Yaron Fairstein, Oren Kalinsky, Zohar Karnin, Guy Kushilevitz, Alexander Libov, and Sofia Tolmach. Class balancing for efficient active learning in imbalanced datasets. In *Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII)*, pages 77–86, 2024.

[55] D Ravikumar. *Towards Enhancement of Machine Learning Techniques Using CSE-CIC-IDS2018 Cybersecurity Dataset, 2021*. PhD thesis, Thesis, Rochester Institute of Technology.

[56] Shampa Banik, Trapa Banik, and Shudipta Banik. Intrusion detection system in smart grid-a review. 2023.

[57] Ahmet Nusret Ozalp and Zafer Albayrak. Detecting cyber attacks with high-frequency features using machine learning algorithms. *Acta Polytechnica Hungarica*, 2022.

[58] Francis Akowuah and Fanxin Kong. Real-time adaptive sensor attack detection in autonomous cyber-physical systems. In *2021 IEEE 27th real-time and embedded technology and applications symposium (RTAS)*, pages 237–250. IEEE, 2021.

[59] Qin Lin, Sicco Verwer, Robert Kooij, and Aditya Mathur. Using datasets from industrial control systems for cyber security research and education. In *Critical Information Infrastructures Security: 14th International Conference, CRITIS 2019, Linköping, Sweden, September 23–25, 2019, Revised Selected Papers 14*, pages 122–133. Springer, 2020.

[60] Lydia Tsiami and Christos Makropoulos. Cyber—physical attack detection in water distribution systems with temporal graph convolutional neural networks. *Water*, 13(9):1247, 2021.

[61] IEC TC57. Iec 61850: Communication networks and systems for power utility automation. *International Electrotechnical Commission Std*, 53:54, 2010.

[62] Munir Majdalawieh, Francesco Parisi-Presicce, and Duminda Wijesekera. Dnpsec: Distributed network protocol version 3 (dnp3) security framework. *Advances in Computer, Information, and Systems Sciences, and Engineering*, 1:227–234, 2006.

[63] National Institute of Standards and Technology. Situational awareness for electric utilities. Technical Report Special Publication 1800-7, NIST, 2019. URL https://www.nccoe.nist.gov/projects/use-cases/situational-awareness.

[64] Carson Hubbard et al. Ten strategies of a world-class cybersecurity operations center. Technical report, MITRE Corporation, 2014. URL https://www.mitre.org/publications/technical-papers/ten-strategies-of-a-world-class-cybersecurity-operations-center.

[65] Wenjie Lu, Jiazheng Li, Yifan Li, Aijun Sun, and Jingyang Wang. A cnn-lstm-based model to forecast stock prices. *Complexity*, 2020(1):6622927, 2020.

[66] Paul Cichonski, Tom Millar, Tim Grance, and Karen Scarfone. Computer Security Incident Handling Guide. Special Publication 800-61 Revision 2, National Institute of Standards and Technology (NIST), August 2012. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf.

[67] Ieee standard for synchrophasor measurements for power systems. *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pages 1–61, 2011. doi: 10.1109/IEEESTD.2011.6111219.

[68] Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. Guide to industrial control systems (ics) security. Technical Report NIST SP 800-82 Rev. 2, National Institute of Standards and Technology, 2015. URL https://doi.org/10.6028/NIST.SP.800-82r2.